AD NUMBER

AD850932

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies and their contractors; Foreign Government Information; DEC 1968. Other requests shall be referred to Army Research Office, Durham, NC.

AUTHORITY

ARO ltr 4 Aug 1971

ARO-D Report 68-3

PROCEEDINGS OF THE 1968 ARMY NUMERICAL ANALYSIS CONFERENCE

Sponsored by
The Army Mathematics Steering Committee
on Behalf of
THE OFFICE OF THE CHIEF OF RESEARCH AND DEVELOPMENT

U.S. Army Research Office-Durham

Report No. 68-3

December 1968

PROCEEDINGS OF THE 1968 ARMY NUMERICAL ANALYSIS CONFERENCE

Sponsored by the Army Mathematics Steering Committee

Host

U.S. Army Electronics Command
Fort Monmouth, New Jersey

25-26 April 1968

U.S. Army Research Office-Durham
Box CM, Duke Station
Durham, North Carolina

FOREWORD

The 1968 Army Numerical Analysis Conference, sponsored by the Army Mathematics Steering Committee (AMSC), was held on 25-26 April 1968 at the U.S. Army Electronics Command, Fort Monmouth, New Jersey. This meeting was the seventh in this series of similar meetings. The main objectives of said conferences are: (1) the exchange of information of interest to the scientific and technical staffs and also the technical managers of the Army "other than business" computer and (2) to provide the AMSC's Subcommittee on Numerical Analysis and Computers with information regarding the Army's current needs in numerical analysis and related fields of mathematics.

Mr. A.D. Hall of Bell Telephone Laboratories was the first of the three invited speakers to give his address. His topic was "An Introduction to ALTRAN". The other two guest speakers were Mr. Gordon Sande from Princeton University and Professor Andries van Dam of Brown University. Respective titles for their addresses were "The Fast Fourier Transforms -- Survey of Selected Applications" and "Computer Graphics". In addition to these informative talks there were eight interesting papers contributed by Army scientists.

Mr. Leon Leskowitz served as Chairman on Local Arrangements. Those in attendance would like to thank him for his efforts in their behalf. Not only did he provide excellent local accommodations, but he also organized a large portion of the program.

Dr. John H. Giese, Chairman of the Conference, has asked that the Proceedings of this meeting be published and issued to interested Army scientists. He wishes to take this occasion to thank all the speakers for their very interesting papers and the various chairmen for their help in the conduction of this meeting.

TABLE OF CONTENTS

*This paper was presented at the Conference. It does not appear in the Proceedings.

UNITED STATES ARMY ELECTRONICS COMMAND
FORT MONMOUTH, NEW JERSEY

PROGRAM
for

1968 ARMY NUMERICAL ANALYSIS CONFERENCE

25-26 April 1968

Thursday, 25 April 1968

TIME	EVENT	PLACE	
0830-0845	Travel via special buses to USAECOM, Hexagon Bldg. Main Entrance	Holiday Inn	Second Lieutenant Robert M. Ireland, Second Lieutenant Ronald R. Shelby, USAECOM Escort Officers
0845-0915	REGISTRATION:	Hexagon Lobby	Registration personnel
	Enroute to Conference Room 1B204	-	USAECOM Escort Officers
0915-0920	WELCOME	Room 1B204	Brigadier General K.M. Gonseth Deputy Commanding General for Operations, USAECOM
0920-0940	OPENING OF THE CONFERENCE:	Room 1B204	Mr. Leon Leskowitz, Chairman on Local Arrangements, USAECOM
0940-1200	SESSION I:	Room 1B204	Mr. Joseph Weinstein, Chairman, USAECOM
0940-1040	An Introduction to ALTRAN	Room 1B204	Mr. A.D. Hall Bell Telephone Laboratories, Murray Hill, N.J.
1040-1110	Coffee Break		
1110-1200	Inversion of Hankel and Toeplitz Matrices	Room 1B204	Mr. Choong Yun Cho, Watervliet Arsenal, Watervliet, New York

Thursday, 25 April 1968

TIME	EVENT	PLACE	
1200-1300	Luncheon		
1300-1800	SESSION II:		
1300-1310	Announcements of future Conferences and Orientation Lectures to be held at the Mathematics Research Center	1B204	Mr. Louis B. Rall, Chairman Mathematics Research Center US Army, University of Wisconsin
1310-1410	Fast Fourier Transforms--Survey of Selected Applications	1B204	Mr. Gordon Sande Math Department, Princeton University Princeton, New Jersey
1410-1440	Coffee break		
1440-1525	Fourier Processing Shortcuts	1B204	Mr. Jerrold Shapiro and Mr. John A. D'Agostino USAECOM
1525-1610	Numerical Experiments on a Stefan Problem	1B204	Mr. Harold J. Breaux, US Army Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland
1610-1620	Break		
1620-1700	Programming for Accuracy	1B204	Mr. J.M. Yohe Mathematics Research Center, US Army, University of Wisconsin
1700-1800	Time Sharing at Fort Monmouth	1B204	Mr. Leon Leskowitz, USAECOM
1800-1815	Bus departs Hexagon and travels to Holiday Inn		USAECOM Escort Officers

Friday, 26 April 1968

0750-0815	Bus departs Holiday Inn; Travel to Hexagon		USAECOM Escort Officers
0815-1210	SESSION III:	1B204	Mr. Sylvan H. Eisman, Chairman, Frankford Arsenal, Philadelphia

Friday, 26 April 1968

TIME	EVENT	PLACE	
0815-0900	Computation of the Dynamic Response of Membranes	1B204	Mr. A.R. Celmins, US Army Ballistic Research Laboratories, Aberdeen Proving Ground, Maryland
0900-0950	Integration of the Time-Dependent Navier-Stokes Equations in Two Dimensions	1B204	Mr. S.C.R. Dennis, Mathematics Research Center, US Army University of Wisconsin Madison, Wisconsin and University of Western Ontario; London, Ontario, Canada
0950-1015	Coffee Break		
1015-1050	Tactical Avionics Systems Simulator	1B204	Mr. Henry Chambers, USAECOM
1050-1210	Computer Graphics	1B204	Professor Andries van Dam Brown University Providence, Rhode Island
1230-	Bus departs Hexagon to Red Bank Airport, bus and train stations		USAECOM Escort Officers

INVERSION OF HANKEL AND TOEPLITZ MATRICES

Choong Yun Cho
Maggs Research Center, Watervliet Arsenal
Watervliet, New York

1. Introduction. In this paper, inversion of the Hankel and Toeplitz matrices is demonstrated in two ways. In the first, we decompose a general matrix into triangular factors expressed in determinantal forms. The inverse of these factors is also obtained in determinantal forms. The results are then applied to the Cauchy matrix, as well as the Hankel and Toeplitz matrices, and explicit expressions for the factors are obtained.

In the second approach, we show that the triangular decomposition of the inverse of the Hankel matrix yields a system of orthogonal polynomials. Then the theory of continued fractions and the qd algorithm for the triangularization of the inverse of the Hankel matrix, are utilized.

A comparison with a standard algorithm for the inversion of the Hankel matrix is made by using the well-known Hilbert matrix.

The Hankel matrix of order (n+1) is of the form

$$H = \begin{bmatrix} c_0 & c_1 & \cdots & c_n \\ c_1 & c_2 & \cdots & c_{n+1} \\ & \cdot & & \\ & \cdot & & \\ & \cdot & & \\ c_n & c_{n+1} & \cdots & c_{2n} \end{bmatrix}$$

$$= ((c_{i+j} \ : \ i, j=0,1,2,\ldots,n))$$

The remainder of this paper was reproduced photographically from the author's manuscript.

Matrices of this form are closely connected to the moment problem on the real axis and also arise as coefficient matrices of the normal equations in problems of least squares polynomial curve fitting.

The Toeplitz matrix of order (n+1) has the form

$$T = \begin{pmatrix} a_0 & a_{-1} & \cdots & a_{-n} \\ a_1 & a_0 & \cdots & a_{-n+1} \\ & \vdots & & \\ a_n & a_{n-1} & \cdots & a_0 \end{pmatrix}$$

$$= ((a_{i-j} \; : \; i, \; j=0,1,2,\ldots,n))$$

such matrices arise in the trigonometric moment problem.

The Toeplitz matrix is closely connected with the Hankel matrix and in numerical computation of the inverse, the algorithm for the Hankel matrix can be carried out exactly in the same way for the Toeplitz matrix.

To be more specific, we consider a Toeplitz matrix $T = ((a_{i-j} \; : \; i,j=0,1,2,\ldots,n))$ of order (n+1) and put $a_{m-n}=b_m$, $m = 0,1,2,\ldots,2n$, then the matrix T becomes

$$T = \begin{pmatrix} b_n & \cdots & b_1 & b_0 \\ b_{n+1} & \cdots & b_2 & b_1 \\ & & \vdots & \\ b_{2n} & \cdots & b_{n+1} & b_n \end{pmatrix}$$

or

$$T = \begin{pmatrix} b_0 & b_1 & \cdots & b_n \\ b_1 & b_2 & \cdots & b_{n+1} \\ \vdots & & & \\ b_n & b_{n+1} & \cdots & b_{2n} \end{pmatrix} \begin{pmatrix} & & & 1 \\ & & \iddots & \\ & 1 & \iddots & \\ 1 & & & \end{pmatrix}$$

Denote $T = H \cdot E$, where E is the elementary matrix, $E = \begin{pmatrix} & & & 1 \\ & & \iddots & \\ & 1 & \iddots & \\ 1 & & & \end{pmatrix}$,

and H a Hankel matrix $((b_{i+j} : i,j=0,1,2,\ldots,n))$.

Then we have $T^{-1} = E \cdot H^{-1}$.

That is, the inverse of Toeplitz matrix can be obtained by inverting the corresponding Hankel matrix and rearranging its rows.

2. Triangular Decomposition of a General Matrix

We first develop the triangularization of a general square matrix in this section and it will be applied to the Cauchy matrix in the following section.

It is convenient to use the following notation for the general square matrix A of order k:

$$A = (a_1\ b_2\ c_3 \cdots v_{k-1}\ w_k)$$

$$= \begin{pmatrix} a_1 & b_1 & c_1 & \cdots & w_1 \\ a_2 & b_2 & c_2 & \cdots & w_2 \\ & & \vdots & & \\ a_k & b_k & c_k & \cdots & w_k \end{pmatrix}$$

We denote the determinant of this matrix by $|a_1b_2c_3 \ldots w_k|$. Thus

$$|b_1d_3h_4| = \det \begin{pmatrix} b_1 & d_1 & h_1 \\ b_3 & d_3 & h_3 \\ b_4 & d_4 & h_4 \end{pmatrix} .$$

It is known (Turnbull [5], p. 369) that for the usual triangular decomposition $A = LU$, the elements of L and U can be expressed in terms of determinants involving the elements of A as follows (it is assumed that the decomposition is possible and L has been chosen to have units in the principal diagonal):

$$L = \begin{pmatrix} 1 & & & & & \\ \frac{|a_2|}{|a_1|} & 1 & & & & \\ \frac{|a_3|}{|a_1|} & \frac{|a_1b_3|}{|a_1b_2|} & 1 & & & \\ & \cdots & & & & \\ \frac{|a_m|}{|a_1|} & \frac{|a_1b_m|}{|a_1b_2|} & \frac{|a_1b_2c_m|}{|a_1b_2c_3|} & \cdots & 1 & \\ & \cdots & & & & \\ \frac{|a_k|}{|a_1|} & \frac{|a_1b_k|}{|a_1b_2|} & \frac{|a_1b_2c_k|}{|a_1b_2c_3|} & \frac{|a_1b_2c_3d_k|}{|a_1b_2c_3d_4|} & \cdots & 1 \end{pmatrix} \qquad (1)$$

and

$$U = \begin{pmatrix} |a_1| & |b_1| & |c_1| & |d_1| & \cdots & |w_1| \\ & \frac{|a_1b_2|}{|a_1|} & \frac{|a_1c_2|}{|a_1|} & \frac{|a_1d_2|}{|a_1|} & \cdots & \frac{|a_1w_2|}{|a_1|} \\ & & \frac{|a_1b_2c_3|}{|a_1b_2|} & \frac{|a_1b_2d_3|}{|a_1b_2|} & \cdots & \frac{|a_1b_2w_3|}{|a_1b_2|} \\ & & & & & \vdots \\ & & & & & \frac{|a_1b_2 \cdots v_{k-1}w_k|}{|a_1b_2 \cdots v_{k-1}|} \end{pmatrix}. \qquad (2)$$

It can be shown (see, Cho[1]) that L^{-1} and U^{-1} can similarly be expressed explicitly as follows:

$$L^{-1} = \begin{pmatrix} 1 & & & \\ -\frac{|a_2|}{|a_1|} & 1 & & \\ \frac{|a_2 b_3|}{|a_1 b_2|} & -\frac{|a_1 b_3|}{|a_1 b_2|} & 1 & \\ -\frac{|a_2 b_3 c_4|}{|a_1 b_2 c_3|} & \frac{|a_1 b_3 c_4|}{|a_1 b_2 c_3|} & -\frac{|a_1 b_2 c_4|}{|a_1 b_2 c_3|} & 1 \\ & \vdots & & \\ (-1)^{k-1} \frac{|a_2 b_3 \cdots v_k|}{|a_1 b_2 \cdots v_{k-1}|} & (-1)^k \frac{|a_1 b_3 c_4 \cdots v_k|}{|a_1 b_2 c_3 \cdots v_{k-1}|} & \cdots & 1 \end{pmatrix}$$

$$\cdots \quad (3)$$

and

$$U^{-1} = \begin{pmatrix} \frac{|1|}{|a_1|} & -\frac{|b_1|}{|a_1b_2|} & \frac{|b_1c_2|}{|a_1b_2c_3|} & -\frac{|b_1c_2d_3|}{|a_1b_2c_3d_4|} & \cdots & (-1)^{k-1}\frac{|b_1c_2d_3\cdots w_{k-1}|}{|a_1b_2c_3\cdots w_k|} \\ & \frac{|a_1|}{|a_1b_2|} & -\frac{|a_1c_2|}{|a_1b_2c_3|} & \frac{|a_1c_2d_3|}{|a_1b_2c_3d_4|} & \cdots & (-1)^{k}\frac{|a_1c_2d_3\cdots w_{k-1}|}{|a_1b_2c_3\cdots w_k|} \\ & & \frac{|a_1b_2|}{|a_1b_2c_3|} & -\frac{|a_1b_2d_3|}{|a_1b_2c_3d_4|} & \cdots & (-1)^{k+1}\frac{|a_1b_2d_3\cdots w_{k-1}|}{|a_1b_2c_3\cdots w_k|} \\ & & & & & \vdots \\ & & & & & \frac{|a_1b_2c_3\cdots v_{k-1}|}{|a_1b_2c_3\cdots w_k|} \end{pmatrix}$$

... (4)

3. Triangular Decomposition of Cauchy Matrices

Recently Schechter [2], and (apparently independently) Trench and Scheinok [4] have shown the elements of the inverse of a Cauchy matrix may be written down in simple closed form.

From the new determinantal expressions that have been developed in Section 2, we now can express the triangular decomposition of the Cauchy matrix explicitly, and furthermore that the same is true of the inverse of these triangular factors.

We denote the Cauchy matrix of order k

$$C = \begin{pmatrix} \frac{1}{\alpha_1+\beta_1} & \frac{1}{\alpha_1+\beta_2} & \cdots & \frac{1}{\alpha_1+\beta_k} \\ \frac{1}{\alpha_2+\beta_1} & \frac{1}{\alpha_2+\beta_2} & \cdots & \frac{1}{\alpha_2+\beta_k} \\ \vdots & & & \\ \frac{1}{\alpha_k+\beta_1} & \frac{1}{\alpha_k+\beta_2} & \cdots & \frac{1}{\alpha_k+\beta_k} \end{pmatrix}$$

by $\left(\left(\frac{1}{\alpha_m+\beta_n}\ ;\ m, n = 1,2,\ldots,k\right)\right)$.

Theorem 1

Given the Cauchy matrix of order k, $C = \left(\left(\frac{1}{\alpha_m + \beta_n}\right); m, n = 1,2,\ldots,k\right)$, where $\alpha_1, \alpha_2, \ldots, \alpha_k; -\beta_1, -\beta_2, \ldots, -\beta_k$, are pairwise distinct and C = LU. Then the triangular factors are

$$L_{m,n} = \frac{(\alpha_n + \beta_n)}{(\alpha_m + \beta_n)} \cdot \prod_{s=1}^{n-1} \frac{(\alpha_n + \beta_s)(\alpha_m - \alpha_s)}{(\alpha_m + \beta_s)(\alpha_n - \alpha_s)} \tag{1}$$

$$U_{m,n} = \frac{1}{(\alpha_m + \beta_n)} \cdot \prod_{s=1}^{m-1} \frac{(\alpha_m - \alpha_s)(\beta_n - \beta_s)}{(\alpha_s + \beta_n)(\alpha_m + \beta_s)} \tag{2}$$

Proof: (see, Cho [1]).

Theorem 2

The elements of the matrix inverses of L and U, where C = LU, are given by

$$(L^{-1})_{m,n} = \prod_{\substack{t=1 \\ t \neq n}}^{m} \frac{1}{(\alpha_n - \alpha_t)} \cdot \prod_{s=1}^{m-1} \frac{(\alpha_m - \alpha_s)(\alpha_n + \beta_s)}{(\alpha_m + \beta_s)} \qquad (m \geq n) \tag{3}$$

$$(U^{-1})_{m,n} = \frac{(\alpha_n + \beta_n)}{\prod_{\substack{t=1 \\ t \neq m}}^{n} (\beta_m - \beta_t)} \cdot \prod_{s=1}^{n-1} \frac{(\alpha_s + \beta_m)(\alpha_n + \beta_s)}{(\alpha_n - \alpha_s)} \qquad (m \leq n) \tag{4}$$

4. Hankel and Toeplitz Matrices

In two special cases it turns out that the Cauchy matrix reduces to one of Toeplitz form: These occur when

$$T_1 = \left(\left(\frac{1}{m - n + x} ; m, n = 1,2,\ldots,k \right) \right) \tag{1}$$

where x is a constant, and

$$T_2 = \left(\left(\frac{1}{p - q^{m-n+x}} ; m, n = 1,2,\ldots,k \right) \right) \tag{2}$$

where p, q, and x are constants.

In the first case we obtain readily L, U, L^{-1}, and U^{-1} by putting $\alpha_m = m$ and $\beta_n = x - n$. The inverse becomes

$$(T_1^{-1})_{m,n} = \frac{\prod_{t=1}^{k} (x + t - m)(x - t + n)}{(n - m + x) \cdot \prod_{\substack{r=1 \\ r \neq m}}^{k} (m - r) \cdot \prod_{\substack{s=1 \\ s \neq n}}^{k} (s - n)} \tag{3}$$

In the second case we note that

$$T_2 = \left(\left(\frac{1}{pq^{n} - q^{m+x}} ; m, n = 1,2,\ldots,k \right) \right) \cdot D$$

where D is a diagonal matrix with $d_{nn} = q^n$, $n = 1,2,\ldots,k$.

The first factor of T_2 now has the Cauchy form, i.e., $\alpha_m = -q^{m+x}$ and $\beta_n = pq^n$, and the triangular factors are readily expressed. The inverse of T_2 is

$$(T_2^{-1})_{m,n} = \frac{\prod_{t=1}^{k} (pq^m - q^{x+t})(pq^t - q^{x+n})}{y \cdot \prod_{\substack{r=1 \\ r \neq m}}^{k} (q^r - q^m) \cdot \prod_{\substack{s=1 \\ s \neq n}}^{k} (q^n - q^s)} \tag{4}$$

where

$$y = p^k q^{kx-x+2m} - p^{k-1} q^{kx+m+n} .$$

The Hankel forms corresponding to the above Toeplitz matrices T_1 and T_2 are

$$H_1 = \left(\left(\frac{1}{m + n + x} ; m, n = 1,2,\ldots,k \right) \right) \tag{5}$$

and

$$H_2 = \left(\left(\frac{1}{p - q^{m+n+x}} ; m, n = 1,2,\ldots,k \right) \right), \tag{6}$$

respectively.

The triangular factors of H_1 and H_2, and their inverses

are immediately obtained by choosing α_n and β_n in the theorems. In particular, the inverse of H_2 is

$$(H_2^{-1})_{m,n} = \frac{\prod_{t=1}^{k} (pq^{-m} - q^{x+t})(pq^{-t} - q^{x+n})}{y \cdot \prod_{\substack{r=1 \\ r\neq m}}^{k} (q^{-r} - q^{-m}) \cdot \prod_{\substack{s=1 \\ s\neq n}}^{k} (q^{n} - q^{s})} \qquad (7)$$

where

$$y = p^k q^{kx-x-2m} - p^{k-1} q^{kx-m+n} .$$

5. The Hankel Matrix In Connection With The Theory of Continued Fractions and Orthogonal Polynomials.

In the previous section we had obtained the triangular factors and the inverse of the Hankel and Toeplitz matrices by triangularization of the Cauchy matrix. This was accomplished using the results for a general matrix expressed in terms of determinantal forms.

We now explore the LU-decomposition of the inverse of the Hankel matrix from a somewhat different point of view.

Suppose we have decomposed a given Hankel matrix H into the LU form where L is unit lower triangular matrix, i.e.,

$$H = LU \tag{1}$$

or
$$L^{-1} H = U \tag{2}$$

Denoting $L^{-1} = K$, we have

$$KH = U$$

The requirement that the product KH be upper triangular imposes (r+ 1) conditions on the (r+ 1)th row of KH, i.e.,

$$\begin{aligned} &\sum_{s=0}^{r} k_{rs}\, c_{s+t} = 0 \qquad t = 0,1,2,\ldots,r-1 \\ &\sum_{s=0}^{r} k_{rs}\, c_{s+r} \neq 0 \qquad (t=r) \end{aligned} \tag{3}$$

where

$$K = \begin{pmatrix} k_{00} & & & \\ k_{10} & k_{11} & & \\ k_{20} & k_{21} & k_{22} & \\ & \vdots & & \end{pmatrix} . \tag{4}$$

Now we define a (linear) functional I such that

$$I\{x^s\} = c_s \tag{5}$$

(e.g., $\int_a^b x^s\, w(x)\, dx = c_s$; the specific forms of $w(x)$ and the interval $[a, b]$ are immaterial at this moment).

Then

$$\sum_{s=0}^{r} k_{rs}\, c_{s+t} = \sum_{s=0}^{r} k_{rs}\, I\{x^{s+t}\}$$

$$= I\left\{x^t \sum_{s=0}^{r} k_{rs}\, x^s\right\}$$

$$= \begin{cases} 0 & t = 0,1,2,\ldots,r-1 \\ \neq 0 & t = r \end{cases} \tag{6}$$

Let $$p_r(x) = \sum_{s=0}^{r} k_{rs}\, x^s \tag{7}$$

then $$I\{x^t\, p_r(x)\} = \begin{cases} 0 & t = 0,1,2,\ldots,r-1 \\ \neq 0 & t = r \end{cases} \tag{8}$$

We may replace x^t by $p_t(x)$ which is a polynomial of degree t, then we have

$$I\{p_t(x)\ p_r(x)\} = \begin{cases} 0 & t = 0,1,2,\ldots,r-1 \\ \neq 0 & t = r \end{cases} \qquad (9)$$

Since t and r are dummy indices, we may write

$$I\{p_t(x)\ p_r(x)\} = \delta_{tr} \cdot d_t \qquad (\ d_t : \text{constant}) \qquad (10)$$

where δ_{tr} is Kronecker delta.

In other words, suppose that from the (r+ 1)th row of the triangular matrix $K = L^{-1}$ defined in Eq. (4), we form the polynomial

$$p_r(x) = \sum_{s=0}^{r} k_{rs}\ x^s$$

Then if we can find a linear functional I such that Eq. (5) is true, this system of polynomials is orthogonal in the sense of Eq. (10).

It is a well-known fact (Tchebychef; Stieltjes) that a power series

$$\sum_{s=0}^{\infty} \frac{c_s}{x^{s+1}} \qquad (11)$$

may be expressed as a continued fraction, the "corresponding" continued fraction which has the form

$$\frac{c_0}{x-}\ \frac{q_1}{1-}\ \frac{e_1}{x-}\ \frac{q_2}{1-}\ \frac{e_2}{x-}\ \cdots \qquad . \qquad (12)$$

The even part of this fraction, the "associated" continued fraction is

$$\frac{c_0}{x-\alpha_0-}\ \frac{\beta_0}{x-\alpha_1-}\ \frac{\beta_1}{x-\alpha_2-}\ \cdots \qquad . \qquad (13)$$

where the connection between the α_r , β_r , and the e_r , q_r is explained later.

It is also well known that both the numerators and denominators of the successive convergents of the associated continued fraction form systems of orthogonal polynomials (see, e.g., Shohat and Sherman [3]; Wynn [7]). The construction of the quantities, e_r , q_r , α_r , and β_r is eased by the use of the q-d algorithm, which we will describe shortly.

Denote the rth convergent of the associated continued fraction by $o_r(x) / p_r(x)$, then $p_r(x)$ is a polynomial of degree r and indeed

$$p_r(x) = \sum_{s=0}^{r} k_{rs}\, x^s$$

where k_{rs} are the elements of the matrix K.

In short, for a given Hankel matrix

$$H = ((c_{i+j} \quad ; i, j = 0,1,2,\ldots))$$

the system of orthogonal polynomials may be constructed via the associated continued fraction. This can be carried out by the q-d algorithm. The process then leads to the matrix K, which is the inverse of the unit lower triangular factor of the given Hankel matrix H.

Furthermore,

$$H = LU = LDL^T \qquad \text{(H symmetric)}$$

and $$H^{-1} = (L^T)^{-1} D^{-1} L^{-1}$$

or $$H^{-1} = K^T D^{-1} K \tag{14}$$

where D is a diagonal matrix with elements d_r , $r = 0,1,2,\ldots$.

The three-term recursion formula for the polynomials $p_r(x)$ is

$$p_r(x) = (x - \alpha_{r-1})\, p_{r-1}(x) - \beta_{r-2}\, p_{r-2}(x) \,. \tag{15}$$

$$r = 2,3,\ldots$$

Multiplying both sides by x^{r-2} and applying the functional I, we have

$$I\{x^{r-2} p_r(x)\} = I\{x^{r-1} p_{r-1}(x)\} - \alpha_{r-1} I\{x^{r-2} p_{r-1}(x)\}$$

$$-\beta_{r-2} I\{x^{r-2} p_{r-2}(x)\} \quad , r = 2,3,\ldots$$

This gives

$$I\{x^{r-1} p_{r-1}(x)\} = \beta_{r-2} I\{x^{r-2} p_{r-2}(x)\}$$

or $$I\{x^{r-1} p_{r-1}(x)\} = \beta_{r-2}\beta_{r-3} I\{x^{r-3} p_{r-3}(x)\}$$

$$\vdots$$

$$= c_o \prod_{s=0}^{r-2} \beta_s$$

i.e., $$d_{r-1} = c_o \prod_{s=0}^{r-2} \beta_s \qquad r = 2,3,\ldots \ . \tag{16}$$

Since $d_o = c_o$, we have obtained a formula for obtaining d_r , $r = 0,1,2,\ldots$. We see readily that $d_r^{1/2}$, $r = 0,1,2,\ldots$, are the normalization factors for the polynomials $p_r(x)$.

6. Quotient-Difference (q-d) Algorithm(+)

(I) In the previous section we considered the power

(+) We base our exposition upon two papers of Wynn ([6], [7])

series

$$\sum_{s=0}^{\infty} \frac{c_s}{x^{s+1}} \,. \tag{1}$$

We could also consider the power series

$$\sum_{s=0}^{\infty} \frac{c_{m+s}}{x^{s+1}} \tag{2}$$

obtained from (1) by removing the first m terms and multiplying the result by x^m.

This is equivalent to taking an arbitrary square section of the Hankel matrix. To indicate this general situation, we write $e_r^{(m)}$ and $q_r^{(m)}$ for e_r and q_r in Section 5.

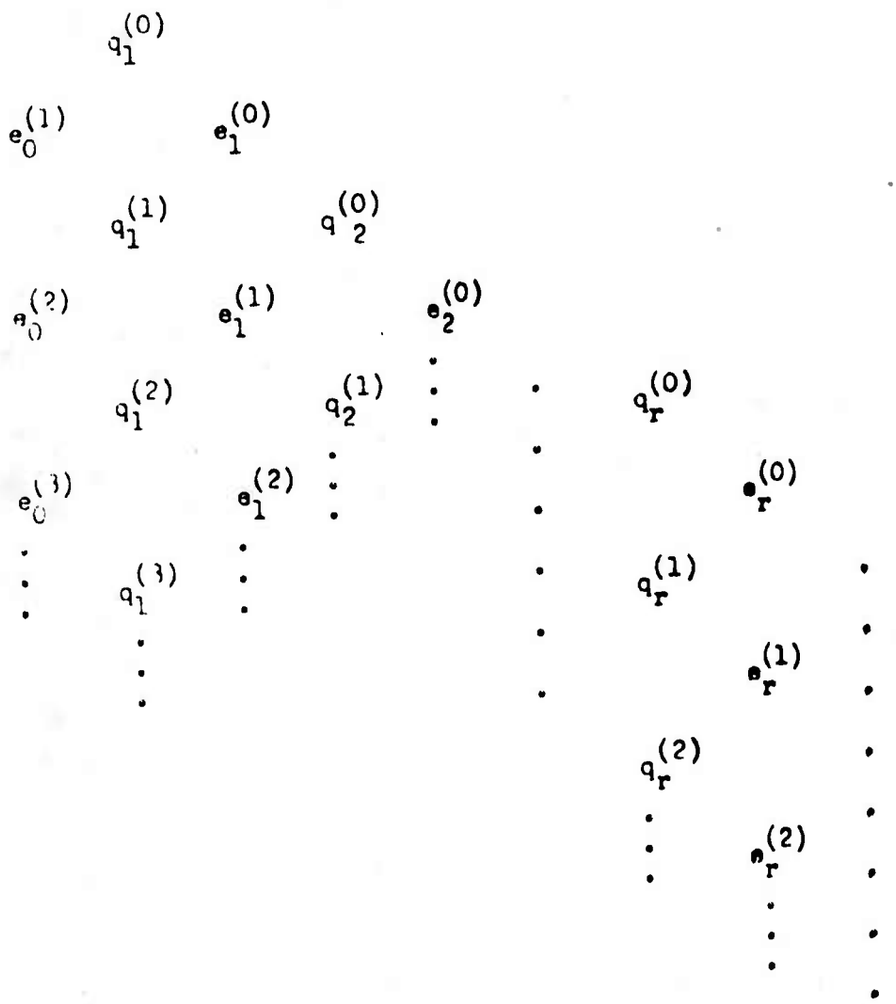
The power series (2) may now be expressed by the corresponding continued fraction

$$\frac{c_m}{x-} \; \frac{q_1^{(m)}}{1-} \; \frac{e_1^{(m)}}{x-} \; \frac{q_2^{(m)}}{1-} \; \frac{e_2^{(m)}}{x-} \cdots \tag{3}$$

The coefficients $q_r^{(m)}$, $e_r^{(m)}$ $(m=0,1,2,\ldots;\ r=1,2,\ldots)$

are obtained by means of the q-d algorithm relationships which were discovered in principle by Stieltjes.

The quantities $q_r^{(m)}$, $e_r^{(m)}$ are placed in the following two dimensional array



We compute $q_r^{(m)}$, $e_r^{(m)}$ by the following relationships

$$e_r^{(m)} = q_r^{(m+1)} - q_r^{(m)} + e_{r-1}^{(m+1)} \tag{4}$$

$$q_r^{(m)} = \frac{q_{r-1}^{(m+1)} \; e_{r-1}^{(m+1)}}{e_{r-1}^{(m)}} \tag{5}$$

with the initial values $e_0^{(m)} = 0 \qquad (m=1,2,\ldots)$

$$q_1^{(m)} = \frac{c_{m+1}}{c_m} \qquad (m=0,1,\ldots) \tag{6}$$

The even part of the continued fraction (3) is

$$\frac{c_m}{x-q_1^{(m)}-} \; \frac{e_1^{(m)} \; q_1^{(m)}}{x-q_2^{(m)} - e_1^{(m)}-} \cdots \quad \frac{e_r^{(m)} \; q_r^{(m)}}{x-q_{r+1}^{(m)} - e_r^{(m)}-} \cdots \tag{7}$$

or

$$\frac{c_m}{x-\alpha_o^{(m)}-} \; \frac{\beta_o^{(m)}}{x-\alpha_1^{(m)}-} \cdots \frac{\beta_{r-2}^{(m)}}{x-\alpha_{r-1}^{(m)}-} \cdots \tag{8}$$

where

$$\alpha_o^{(m)} = q_1^{(m)} \qquad (m=0,1,2,\ldots)$$

$$\beta_{r-2}^{(m)} = e_{r-1}^{(m)} \; q_{r-1}^{(m)} \; , \quad \alpha_{r-1}^{(m)} = q_r^{(m)} + e_{r-1}^{(m)} \tag{9}$$

$$(m=0,1,\ldots;\; r=2,3,\ldots)$$

The rth convergent of expansion (8) may be written as the quotient of two polynomials

$$\frac{o_r^{(m)}(x)}{p_r^{(m)}(x)} \tag{10}$$

where $p_r^{(m)}(x)$ is of degree r and may be written as

$$p_r^{(m)}(x) = \sum_{s=0}^{r} k_{r,s}^{(m)} x^s \tag{11}$$

and $o_r^{(m)}(x)$ is of degree $(r-1)$ so that

$$o_r^{(m)}(x) = \sum_{s=0}^{r-1} i_{r,s}^{(m)} x^s \quad . \tag{12}$$

The polynomials $p_r^{(m)}(x)$ form an orthogonal system in the sense that

$$\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+t+s} = 0 \qquad (t=0,1,2,\ldots,r-1) \tag{13}$$

while

$$\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+r+s} \neq 0 \; . \tag{14}$$

The inequality (14) is rendered definite by imposing that

$$k_{r,r}^{(m)} = 1 \qquad (r=0,1,2,\ldots) \; . \tag{15}$$

Then the polynomials $p_r^{(m)}(x)$ are the denominators of the convergents of expansion (8); they satisfy the three-term recursion formula

$$p_r^{(m)}(x) = (x - \alpha_{r-1}^{(m)})\, p_{r-1}^{(m)}(x) - \beta_{r-2}^{(m)}\, p_{r-2}^{(m)}(x) \qquad (16)$$

$$(r=2,3,\ldots)$$

with

$$p_0^{(m)}(x) = 1\,, \quad p_1^{(m)}(x) = x - \alpha_0^{(m)} \qquad (17)$$

If the coefficients $e_r^{(m)}$, $q_r^{(m)}$, $\alpha_{r-1}^{(m)}$, $\beta_{r-2}^{(m)}$ are available then the polynomials $p_r^{(m)}(x)$ may be constructed by the recursion formula (16).

Relationships (16) and (17) may be expressed in terms of the coefficients $k_{r,s}^{(m)}$, $m=0,1,2,\ldots$,

$$k_{r,s}^{(m)} = k_{r-1,s-1}^{(m)} - \alpha_{r-1}^{(m)} k_{r-1,s}^{(m)} - \beta_{r-2}^{(m)}\, k_{r-2,s}^{(m)} \qquad (18)$$

$$(s=1,2,\ldots,r-2)$$

$$k_{r,r-1}^{(m)} = k_{r-1,r-2}^{(m)} - \alpha_{r-1}^{(m)}$$

$$(r=2,3,\ldots) \qquad (19)$$

$$k_{r,0}^{(m)} = - \alpha_{r-1}^{(m)}\, k_{r-1,0}^{(m)} - \beta_{r-2}^{(m)}\, k_{r-2,0}^{(m)}$$

$$(r=2,3,\ldots)$$

with the initial values $k_{1,0}^{(m)} = - c_{m+1} / c_m$.

We have seen that the coefficients $q_r^{(m)}$, $e_r^{(m)}$, $\alpha_{r-1}^{(m)}$ and $\beta_{r-2}^{(m)}$ may be obtained by the q-d algorithm, i.e., by means of relationships (4), (5), (6) and (9).

With regard to the construction of $\alpha_{r-1}^{(m)}$, $\beta_{r-2}^{(m)}$ we have another method at our disposal - an orthogonal procedure, which is equivalent to the triangular decomposition of a given Hankel matrix.

Denote

$$\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+r+s} = d_r^{(m)} \tag{20}$$

$$\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+r+s+1} = g_r^{(m)} \tag{21}$$

$$k_{r,r} = 1$$

$$(r=0,1,2,\ldots)$$

then the three-term recursion formula (16) yields

$$0 = d_{r-1}^{(m)} - \beta_{r-2}^{(m)} d_{r-2}^{(m)} \tag{22}$$

i.e. $$\beta_{r-2}^{(m)} = \frac{d_{r-1}^{(m)}}{d_{r-2}^{(m)}} \qquad (r=2,3,\ldots) \tag{23}$$

and

$$0 = g_{r-1}^{(m)} - \alpha_{r-1}^{(m)} d_{r-1}^{(m)} - \beta_{r-2}^{(m)} g_{r-2}^{(m)} \tag{24}$$

i.e. $$\alpha_{r-1}^{(m)} = \frac{g_{r-1}^{(m)} - \beta_{r-2}^{(m)} g_{r-2}^{(m)}}{d_{r-1}^{(m)}} \tag{25}$$

or $$\alpha_{r-1}^{(m)} = \frac{g_{r-1}^{(m)}}{d_{r-1}^{(m)}} - \frac{g_{r-2}^{(m)}}{d_{r-2}^{(m)}} \tag{26}$$

$$(r=2,3,\ldots)$$

From equation (17), we have

$$\alpha_0^{(m)} = -k_{1,0}^{(m)} \ . \tag{27}$$

Now we define quantity $h_{t,r}^{(m)}$ by

$$h_{t,r}^{(m)} = \frac{\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+t+s}}{\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+r+s}} \tag{28}$$

or $$h_{t,r}^{(m)} = \frac{\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+t+s}}{d_r^{(m)}} \tag{29}$$

$$(r=0,1,2,\ldots,t-1) .$$

Then in particular

$$h^{(m)}_{r+1,r} = \frac{g^{(m)}_r}{d^{(m)}_r} \qquad (r=0,1,2,\ldots) \tag{30}$$

and equation (26) becomes

$$\alpha^{(m)}_{r-1} = h^{(m)}_{r,r-1} - h^{(m)}_{r-1,r-2} \tag{31}$$

$$(r=2,3,\ldots)\,.$$

It can be shown that

$$d^{(m)}_t = c_{m+2t} - \sum_{r=0}^{t-1} d^{(m)}_r \cdot \left(h^{(m)}_{t,r}\right)^2 \tag{32}$$

$$(t=1,2,\ldots)$$

with

$$d^{(m)}_0 = c_m \tag{33}$$

and

$$k^{(m)}_{t,s} = -\sum_{r=s}^{t-1} h^{(m)}_{t,r}\, k^{(m)}_{r,s} \tag{34}$$

$$(s=0,1,2,\ldots,t-1).$$

Thus equations (23) to (34) offer the following recursive procedure.

Starting with (i.e. t=0)

$$d^{(m)}_0 = c_m \;;\; h^{(m)}_{0,0} = 1 \;;\; k^{(m)}_{0,0} = 1$$

we form (t=1)

$$h_{1,0}^{(m)} = c_{m+1} / c_m$$

$$k_{1,0}^{(m)} = - h_{1,0}^{(m)}$$

$$d_1^{(m)} = c_{m+2} - d_0^{(m)} \cdot h_{1,0}^{(m)2}$$

and obtain

$$\beta_0^{(m)} = d_1^{(m)} / d_0^{(m)}$$

$$\alpha_0^{(m)} = - k_{1,0}^{(m)} \qquad \text{(Equation (27))} .$$

For t=2 we have

$$h_{2,0}^{(m)} = c_{m+2} / d_0^{(m)}$$

$$h_{2,1}^{(m)} = (c_{m+3} + k_{1,0}^{(m)} c_{m+2}) / d_1^{(m)}$$

$$k_{2,0}^{(m)} = - (h_{2,0}^{(m)} + h_{2,1}^{(m)} k_{1,0}^{(m)})$$

$$k_{2,1}^{(m)} = - h_{2,1}^{(m)}$$

$$d_2^{(m)} = c_{m+4} - d_0^{(m)} \cdot h_{2,0}^{(m)2} - d_1^{(m)} \cdot h_{2,1}^{(m)2}$$

we obtain

$$\alpha_1^{(m)} = h_{2,1}^{(m)} - h_{1,0}^{(m)}$$

$$\beta_1^{(m)} = d_2^{(m)} / d_1^{(m)}$$

We repeat the steps for t=3,4,...,n .

Before we show a relationship among $d_r^{(m)}$, $h_{r,s}^{(m)}$, $k_{r,s}^{(m)}$ and a given Hankel matrix, we note that equations (29) and (34) yield

$$k_{t,s}^{(m)} = -\sum_{r=s}^{t-1} k_{r,s}^{(m)} \left[\frac{\sum_{u=0}^{r} k_{r,u}^{(m)} c_{m+t+u}}{d_r^{(m)}} \right] \qquad (35)$$

$$(s=0,1,2,\ldots,t-1)$$

and from equations (29) and (32) we have

$$d_t^{(m)} = c_{m+2t} - \sum_{r=0}^{t-1} \frac{\left[\sum_{s=0}^{r} k_{r,s}^{(m)} c_{m+t+s} \right]^2}{d_r^{(m)}} \qquad (36)$$

$$(t=1,2,\ldots) .$$

By expression (23) we see that

$$d_t^{(m)} = c_m \prod_{r=0}^{t-1} \beta_r^{(m)} \qquad (t=1,2,\ldots) \qquad (37)$$

$$d_0^{(m)} = c_m .$$

(II) To summarize, we denote

$$K^{(m)} = \begin{bmatrix} k_{0,0}^{(m)} & & & \\ k_{1,0}^{(m)} & k_{1,1}^{(m)} & & \\ \vdots & & & \\ k_{n,0}^{(m)} & k_{n,1}^{(m)} & \cdots & k_{n,n}^{(m)} \end{bmatrix} \tag{38}$$

$$L^{(m)} = \begin{bmatrix} h_{0,0}^{(m)} & & & \\ h_{1,0}^{(m)} & h_{1,1}^{(m)} & & \\ \vdots & & & \\ h_{n,0}^{(m)} & h_{n,1}^{(m)} & \cdots & h_{n,n}^{(m)} \end{bmatrix} \tag{39}$$

$$D^{(m)} = \begin{bmatrix} d_0^{(m)} & & & \\ & d_1^{(m)} & & \\ & & \ddots & \\ & & & d_n^{(m)} \end{bmatrix} \tag{40}$$

where $k_{r,s}^{(m)}$, $h_{r,s}^{(m)}$ and $d_r^{(m)}$ are defined in the previous paragraph.

For a given Hankel matrix of order n+1

$$H^{(m)} = \begin{bmatrix} c_m & c_{m+1} & \cdots & c_{m+n} \\ c_{m+1} & c_{m+2} & \cdots & c_{m+n+1} \\ & \vdots & & \\ c_{m+n} & c_{m+n+1} & \cdots & c_{m+2n} \end{bmatrix} \tag{41}$$

expressions (35), (36), (37), (40) and (41) may be assembled as the matrix equation

$$K^{(m)} \; H^{(m)} \; K^{(m)T} = D^{(m)} \tag{42}$$

In fact

$$K^{(m)} \; L^{(m)} = L^{(m)} \; K^{(m)} = I \tag{43}$$

and we have

$$H^{(m)} = L^{(m)} \; D^{(m)} \; L^{(m)T} \tag{44}$$

$$H^{(m)^{-1}} = K^{(m)T} \; D^{(m)^{-1}} \; K^{(m)} \tag{45}$$

Relationship (44) is the triangular decomposition of Hankel matrix (41).

In terms of the quantities $d_r^{(m)}$, $k_{r,s}^{(m)}$ the general element of $H^{(m)^{-1}}$ is given by

$$\left[H^{(m)^{-1}}\right]_{i,j} = \sum_{r=\max(i,j)}^{n} k_{r,i}^{(m)} \; d_r^{(m)^{-1}} \; k_{r,j}^{(m)} \tag{46}$$

and in terms of the quantities $k_{r,s}^{(m)}$, $\beta_s^{(m)}$, $\left[H^{(m)^{-1}}\right]_{i,j}$ is given by

$$\left[H^{(m)^{-1}}\right]_{i,j} = c_m^{-1} \sum_{r=\max(i,j)}^{n} \frac{k_{r,i}^{(m)} k_{r,j}^{(m)}}{\prod_{s=0}^{r-1} \beta_s^{(m)}} \qquad (47)$$

$$(0 \leq i,j \leq n)$$

Thus the groups of equations (23) - (34) in conjunction with (46), and equations (4), (5), (6), (9), (16), (17) in conjunction with (47), offer two algorithms for the inversion of the Hankel matrix $H^{(m)}$, an orthogonal procedure and a q-d algorithm, respectively.

An actual comparison for the algorithms will be demonstrated in the following paragraph.

(III) A Comparison Between Exact and Computed Results

These closed formulas enable us in the given cases to compare the exact values of $(H^{-1})_{i,j}$ with those arising from machine computation based upon Gauss elimination and the two methods described in the previous sections.

Since the Hilbert matrix is a form of Hankel matrix, $((c_m = 1/(m+1);\ m=0,1,2,\ldots))$, it is interesting to compare our methods with Gauss method (e.g., Crout reduction with pivoting) for the inversions of 4x4, 5x5, 6x6, 7x7, 8x8 and 9x9 matrices by a CDC 1604 computer. In the following table we show the three corner elements of each inverse, i.e., (0,0), (0,n) and (n,n) elements, obtained by three different algorithms. Throughout the computation single precisions were used.

TABLE

Row (A) q-d algorithm
Row (B) Crout reduction
Row (C) orthogonal procedure
Row (D) exact value

Order		(0,0)	(0,n)	(n,n)
4	(A)	1.6000000162 E1	-1.4000000260 E2	2.8000000418 E3
	(B)	1.6000000512 E1	-1.4000000838 E2	2.8000001360 E3
	(C)	1.6000000275 E1	-1.4000000407 E2	2.8000000585 E3
	(D)	1.6000000000 E1	-1.4000000000 E2	2.8000000000 E3

Order		(0,0)	(0,n)	(n,n)
5	(A)	2.50000054458 E1	6.30000320214 E2	4.4100018884 E4
	(B)	2.5000012054 E1	6.3000067790 E2	4.4100038862 E4
	(C)	2.5000005115 E1	6.3000027626 E2	4.4100014959 E4
	(D)	2.5000000000 E1	6.3000000000 E2	4.4100000000 E4
6	(A)	3.6000183765 E1	-2.7720379450 E3	6.9855180104 E5
	(B)	3.6000366371 E1	-2.7720752723 E3	6.9855999948 E5
	(C)	3.6000497287 E1	-2.7721155771 E3	6.9857099602 E5
	(D)	3.6000000000 E1	-2.7720000000 E3	6.9854400000 E5
7	(A)	4.9003225713 E1	1.2014192924 E4	1.1100543412 E7
	(B)	4.9010533816 E1	1.2019843974 E4	1.1105073673 E7
	(C)	4.9018531675 E1	1.2025989916 E4	1.1109544587 E7
	(D)	4.9000000000 E1	1.2012000000 E4	1.1099088000 E7
8	(A)	6.4027483768 E1	-5.1541127634 E4	1.7681324615 E8
	(B)	6.4295043433 E1	-5.2301722659 E4	1.7895337516 E8
	(C)	6.3039472898 E1	-4.8104553005 E4	1.6504227376 E8
	(D)	6.4000000000 E1	-5.1480000000 E4	1.7667936000 E8
9	(A)	8.2689617719 E1	2.3841693846 E5	3.0458808566 E9
	(B)	9.2825392880 E1	3.4429869353 E5	4.1271688201 E9
	(C)	6.3366261800 E1	1.1675106743 E5	4.1711366759 E9
	(D)	8.1000000000 E1	2.1879000000 E5	2.8158273000 E9

7. Two Special Cases

In Section 4, we obtained explicit expressions for the inverse of two special matrices, H_1 and H_2 (T_1 and T_2), through the triangularization of the Cauchy matrix. We now show the results via the q-d algorithm and will see where these matrices are derived from.

We have shown in Eq. (47) of Section 6 that

$$\left[H^{(m)^{-1}}\right]_{i,j} = c_m^{-1} \sum_{r=\max(i,j)}^{n} \frac{k_{r,i}^{(m)} k_{r,j}^{(m)}}{\prod_{s=0}^{r-1} \beta_s^{(m)}} \qquad (1)$$

$$(0 \leq i,j \leq n)$$

where $H^{(m)}$ is of order (n+1).

Wynn [7] obtains by the q-d algorithm closed expressions for $e_r^{(m)}$, $q_r^{(m)}$, and $k_{r,s}^{(m)}$ of the following Hankel matrices:

$$H_1 : \begin{cases} c_0 = 1 \\ c_m = \dfrac{\alpha(\alpha+1) \ldots (\alpha+m-1)}{\gamma(\gamma+1) \ldots (\gamma+m-1)} \end{cases} \qquad (m=1,2,\ldots)$$

and

$$H_2 : \begin{cases} c_0 = 1 \\ c_m = \dfrac{(A-q^{\alpha})(A-q^{\alpha+1}) \ldots (A-q^{\alpha+m-1})}{(C-q^{\gamma})(C-q^{\gamma+1}) \ldots (C-q^{\gamma+m-1})} \end{cases} \qquad (m=1,2,\ldots)$$

where A, C, q, α, and γ are constants.

For H_1 we have

$$k_{r,s}^{(m)} = \frac{(-1)^{r-s}\binom{r}{s}\binom{\alpha+m+r-1}{r-s}}{\binom{\gamma+m+2r-2}{r-s}} \qquad (2)$$

$$(m=0,1,2,\ldots;\ r=1,2,\ldots;\ s=0,1,\ldots,r-1),$$

$$\beta_r^{(m)} = \frac{(r+1)(\gamma-\alpha+r)(\alpha+m+r)(\gamma+m+r-1)}{(\gamma+m+2r-1)(\gamma+m+2r)^2(\gamma+m+2r+1)} \qquad (3)$$

$$(r=0,1,2,\ldots)$$

and for H_2

$$k_{r,s}^{(m)} = (-1)^{r-s}\, q^{\frac{(r-s)(r-s-1)}{2}} \frac{(1-q^r)(1-q^{r-1})\ldots(1-q^{r-s+1})}{(1-q)(1-q^2)\ldots(1-q^s)}$$

$$\cdot\frac{(A-q^{\alpha+m+s})(A-q^{\alpha+m+s+1})\ldots(A-q^{\alpha+m+r-1})}{(C-q^{\gamma+m+r+s-1})(C-q^{\gamma+m+r+s})\ldots(C-q^{\gamma+m+2r-2})} \qquad (4)$$

$$(m=0,1,2,\ldots;\ r=1,2,\ldots;\ s=0,1,\ldots,r-1),$$

$$\beta_r^{(m)} = q^{m+2r}\frac{(1-q^{r+1})(Cq^{\alpha}-Aq^{\gamma+r})(A-q^{\alpha+m+r})(C-q^{\gamma+m+r-1})}{(C-q^{\gamma+m+2r-1})(C-q^{\gamma+m+2r})^2(C-q^{\gamma+m+2r+1})} \qquad (5)$$

$$(r=0,1,2,\ldots).$$

Introducing these expressions in Eq.(1), we may obtain $H^{(m)-1}$ in closed forms, i.e., in terms of simple products and exponential factors. It is a curious fact, and one of the results of our

investigation, that these scalar products can be simplified only when $\gamma = \alpha+1$ and $A = C$, namely,

$$H_1 = ((c_m = \frac{\alpha}{\alpha+m} ; m=0,1,2,\ldots,n)) ,$$

and

$$H_2 = ((c_m = \frac{A-q^{\alpha}}{A-q^{\alpha+m}} ; m=0,1,2,\ldots,n)) .$$

The inverse becomes

$$\left[H_1^{(0)^{-1}}\right]_{i,j} = \frac{(-1)^{i+j} \prod_{r=i}^{n+i} (\alpha+r) \prod_{s=j}^{n+j} (\alpha+s)}{i!\, j!\, (n-i)!\, (n-j)!\, \alpha(\alpha+i+j)} \qquad (6)$$

$$(0 \leq i,j \leq n),$$

and

$$\left[H_2^{(0)^{-1}}\right]_{i,j} = \frac{\prod_{t=0}^{n} (A-q^{\alpha+i+t})(A-q^{\alpha+j+t})}{A^n\, q^{n\alpha}\, (A-q^{\alpha})(A-q^{\alpha+i+j}) \prod_{\substack{r=0 \\ r\neq i}}^{n} (q^i-q^r) \prod_{\substack{s=0 \\ s\neq j}}^{n} (q^j-q^s)} \qquad (7)$$

$$(0 \leq i,j \leq n),$$

respectively.

We mentioned in the introduction that there is a simple connection between Hankel and Toeplitz matrices; this enables

immediately to derive the inverse of the following Toeplitz matrices of order (n+1):

$$T_1 = ((a_k = \frac{\alpha}{\alpha+k} \; ; \; -n \leq k \leq n)),$$

and

$$T_2 = ((a_k = \frac{A-q^{\alpha}}{A-q^{\alpha+k}} \; ; \; -n \leq k \leq n))$$

where A, q, and α are constants.

The inverse becomes

$$(T_1^{-1})_{i,j} = \frac{(-1)^{n-i+j} \prod_{k=n-i}^{2n-i} (\alpha-n+k) \prod_{s=j}^{n+j} (\alpha-n+s)}{i!\; j!\; (n-i)!\; (n-j)!\; \alpha(\alpha-i+j)}, \qquad (8)$$

$$(0 \leq i,j \leq n)$$

and

$$(T_2^{-1})_{i,j} = \frac{\prod_{k=0}^{n} (A-q^{\alpha-i+k})(A-q^{\alpha-n+j+k})}{y \cdot \prod_{\substack{r=0 \\ r \neq n-i}}^{n} (q^{n-i}-q^{r}) \prod_{\substack{s=0 \\ s \neq j}}^{n} (q^{j}-q^{s})} \qquad (9)$$

$$\text{where } y = A^n q^{n(\alpha-n)} (A-q^{\alpha})(A-q^{\alpha-i+j})$$

$$(0 \leq i,j \leq n),$$

respectively.

REMARKS :

1. Note that H_2 and T_2 become

$$H_2 \longrightarrow H_1$$

$$T_2 \longrightarrow T_1$$

in the limit as $q \longrightarrow 1$ and $A = C = 1$.

2. In Section 2 through Section 4, elements of the matrices are indexed as $(a_{11}, a_{12}, \ldots)$ instead of $(a_{00}, a_{01}, \ldots)$ which is more conventional for the Hankel and Toeplitz matrices. The formulas appear to be different in two conventions, but they are readily modified from one system to another.

3. Throughout this paper, all the empty products in expressions are assumed to be unity.

8. Acknowledgement

The author is indebted to Professors B. Noble and P. Wynn, Mathematics Research Center, University of Wisconsin, for their guidance and encouragement throughout the project.

REFERENCES

1. Choong Yun Cho, On the Triangular Decomposition of Cauchy Matrices, (to appear in Math. of Comp., October, 1968).
2. S. Schechter, On the Inversion of Certain Matrices, Math. of Comp., vol. 13, 1959, pp. 73-77.
3. J. Shohat and J. Sherman, On the Numerators of the Continued Fraction, Proc. Nat. Acad. Sci., vol. 18, 1932, pp. 283-287.
4. W. F. Trench and P. A. Scheinok, On the Inversion of a Hilbert Type Matrix, SIAM Review, vol. 8, 1966, pp. 57-61.
5. H. W. Turnbull, The Theory of Determinants, Matrices, and Invariants, Dover, New York, 1960.
6. P. Wynn, The Rational Approximation of Functions which are Formally Defined by a Power Series Expansion, Math. of Comp., vol. 14, 1960, pp. 147-186.
7. P. Wynn, Upon a General System of Orthogonal Polynomials, Math. Research Center Technical Summary Report No. 548, University of Wisconsin, March 1965.

BLANK PAGE

FOURIER PROCESSING SHORTCUTS

John D'Agostino and Jerrold M. Shapiro
U.S. Army Electronics Command
Fort Monmouth, New Jersey

INTRODUCTION

The advent of the Fast Fourier Transform[1] (FFT) makes it possible to rapidly transform or correlate a large volume of data.[2] The available FFT algorithms[1,3] accept as input a sequence of discrete complex numbers. When the data to be Fourier processed consist of real points, direct use of the FFT algorithm is wasteful of computer storage space and time. A technique will be described which processes certain redundant types of data with a savings of half the time and half the computer storage required for direct application of the FFT algorithm. The technique does not require modification to available FFT algorithms. Typical of the Fourier processing operations which can be performed at substantial savings are crosscorrelation, autocorrelation, convolution, deconvolution and two-channel Fourier transformation. The theory behind these techniques is presented in the first part, and their implementation is detailed in the second part.

THEORY

The Fast Fourier Transform is a rapid implementation of the finite discrete Fourier Transform, or FDFT. The FDFT of a sequence of real data points is a complex function with symmetry in its real and imaginary parts. This symmetry may be used to effect shortcuts when working with data sequences which are either real or have real FDFT's.

This paper was photographically reproduced from the author's copy.

Before demonstrating this symmetry, let us review some properties of the Fast Fourier Transform algorithms. These algorithms assume that the data sequence forms one cycle of a periodic function. Consequently, the index of the data sequence must be considered as modulo N, where N is the number of points in the sequence. The number of transform values returned by the FFT algorithms is the same as the number of input points. The input points are considered to be equispaced in each dimnesion. In general, the data points are assumed to be complex numbers. This last assumption of FFT algorithms causes the FFT to be slower than necessary when a real data sequence is to be transformed.

The symmetry properties for continuous Fourier Transforms are well known. Those for the FDFT are similar. If a real funciton is placed in the real part of a complex function, and if the imaginary part is set to zero, then the transform of the complex function will be Hermitian symmetric. The real part of the transform will have even symmetry while the imaginary part of the transform will have odd symmetry. If the real function had been placed in the imaginary part of the complex function, and the real part set to zero, then the transform would again be Hermitian symmetric, except that now the real part would have odd symmetry, and the imaginary part would have even symmetry. These symmetry properties are discussed in detail in Appendix A.

If the complex function had been composed of two real functions, one each in its real and imaginary parts, then we see that the transform will no longer be symmetric. However, the real and imaginary parts of the transform may each be separated into their even and odd components. This

process is called symmetrization. Since we know where each symmetry component originated, we can reconstruct the transforms of each of the two real functions we started with. The process of obtaining two spectra with one transform has been described in the literature[4,5], and is here called two-channel Fourier transformation. (See Appendix B) But much more can be done.

The Fourier processing operations listed in the introduction can be performed in the transform space, and involve only point by point multiplications or divisions of the transforms of two real functions. We have seen that all the components of each transform are available after two-channel Fourier transformation. These components may be arithmetically assembled to yield, on a successive transform, the result of the desired processing operation. Since this result is a real function, its transform will have the symmetry previously described. Therefore, the second transform can be a two-channel one, and can produce the results of two processing operations at once. This results in substantial savings over direct application of the FFT algorithm. When two processing operations are carried out at once, only two transform operations are required, while direct application of the algorithm to each real function requires six transform operations. The time for direct application may be reduced to four transform operations, but the computer storage requirement then doubles.

The Fourier processing shortcut is two to three times faster, and uses up to half the storage required by direct application of the FFT. Note that no modification to the FFT algorithm is required but that the increase in speed is a result of the symmetry of the transform of real functions.

The central idea in the following explanations is that the FDFT algorithm causes the symmetry components of a complex function to follow fixed paths, so that a careful choice of the terms put at the start of each path will result in the desired processing operations at the end of the path. The paths, or symmetry properties, are listed below, and demonstrated in Appendix A. The properties are:

A. real even function $\longrightarrow$ real even function

B. real odd function $\longrightarrow$ imaginary odd function

C. imaginary even function $\longrightarrow$ imaginary even function

D. imaginary odd function $\longrightarrow$ real odd function

The arrow indicates that the FDFT algorithm applied to the type of function at its tail produces a function of the type at its head. An imaginary function is defined here as a function whose values are $\sqrt{-1}$ times a real number.

An example of the application of the symmetry properties to two-channel Fourier transformation is given in Appendix B.

In the next section, the Fourier Processes are discussed.

FOURIER PROCESSES

The term Fourier Process refers to the following series of mathematical operations. First, two real functions, X and Y, are each Fourier transformed to obtain their complex transforms, X and Y. Then X and Y are arithmetically operated on and combined to form a single complex function Z according to a prescribed rule. Inverse Fourier transformation of Z produces a specfic real function Q which is the end product of the process. The nature of Q will depend, for the most part, on the manner in which Z is formed from X and Y.

To summarize:

$$Q = \text{FDFT}[Z]$$

where $$Z = Z[X, Y]$$

$$X = \text{FDFT}[X]$$

$$Y = \text{FDFT}[Y]$$

X, Y are real functions

FDFT [-] denotes the finite discrete Fourier transform of the quantity in brackets.

Figure 1 lists the three basic Fourier Processes treated in this paper. Each process is defined mathematically in the column at the left. The transform operation associated with each process is listed to its right. Thus convolution is associated with complex multiplication of X by Y; crosscorrelation with complex multiplication of X by $\overset{*}{Y}$ (where the symbol * over a variable stands for complex conjugate); and deconvolution

with complex division of X by Y. In the last case multiplication of both numerator and denominator by Y* rationalizes the fraction X/Y.

Since deconvolution is not a standard Fourier Process, it will be explained more fully. Deconvolution is, in a sense, the inverse of convolution. X is the function resulting from the convolution of Q with Y. X and Y are both known and Q must be determined. The transform operation required to do this is complex division of X by Y. In the event that Y contains one or more zero-valued elements, the process described by X/Y cannot be accomplished completely and consequently Q will be only partially determined.

PROCESSING SHORTCUTS

Generally, Fast Fourier Transform Algorithms are defined in terms of the operation on complex data. When the input data are completely real, as they are in many cases, this restriction is wasteful. It has been shown that a second real function can be inserted into the otherwise blank imaginary part of the input data to form an overall complex function Z. (see figure 2) After complex Fourier Transformation the separate complex transforms X and Y can be separated from the overall transform Z by the application of a set of simple arithmetic formulae (see figure 3). These formulae are based solely on the symmetry properties of the odd and even components of X and Y (see Appendix A) and on the manner in which these components are embedded in Z (Appendix B). In some cases, however,

it is possible to perform the required operations on X and Y without explicitly extracting these functions from Z.

There are thus two basic types of shortcut techniques. Both types capitalize on the fact that X and Y are completely real by forming a pseudo-complex function Z from them. X forms the real part of Z and Y the imaginary part. However, one type of shortcut enables Fourier Processing of X and Y without explicit determination of their transforms $\hat{X}$ and $\hat{Y}$. The second type of shortcut extracts X and Y from Z and operates on them directly.

THE BASIC OPERATION

All of the shortcuts produce in the transform space the results of the desired multiplication (or division) expressed in terms of the even and odd symmetry components. For example, to get XY in the transform space, the following must be done.

$$\hat{X}\hat{Y} = (\hat{X}_e + i\hat{X}_0)(\hat{Y}_e + i\hat{Y}_0)$$

$$\hat{X}\hat{Y} = \hat{X}_e\hat{Y}_e + \hat{X}_0\hat{Y}_0(i)^2 \qquad \text{even components}$$

$$+i\hat{X}_e\hat{Y}_0 + i\hat{X}_0\hat{Y}_e \qquad \text{odd components}$$

$$\hat{X}\hat{Y} = \hat{X}_e\hat{Y}_e - \hat{X}_0\hat{Y}_0$$

$$+i(\hat{X}_e\hat{Y}_0 + \hat{X}_0\hat{Y}_0)$$

To produce $\hat{\hat{X}}\hat{Y}$, put the even components in the even part of the real part of the transform. These terms will travel to the even part of the real part of the next transform (see figures 2, 4). Put the odd components times (-1) in the odd part of the imaginary part of the transform. These terms will be multiplied by (-1) and then will travel to the odd part of

the real part of the next transform (see figures 2, 4). Thus the term Q of figure 4 is the transform of $\hat{X}\hat{Y}$, that is (see figure 1), Q is X * Y, the convolution function. This example illustrates the basic operation of multiplication in terms of symmetry components.

SHORTCUTS USING IMPLICIT TRANSFORMS

Figure 4 illustrates the first type of shortcut. (X * Y) is determined at Q without extracting X and Y from Z. The asterisk between X and Y denotes the convolution operation. The operations indicated take place after the initial Fourier transformation of Z to obtain $\hat{Z}$. RE and IM designate the real and imaginary parts of $\hat{Z}$ respectively. RE'and IM'are the new $\hat{Z}$ resulting from the designated operations. Even and odd symmetry components of REAL' (same as RE') and IMAGINARY' (same as IM') have been grouped separately with even components in the top half of each box and odd components in the bottom half.

Close inspection of the components which travel to Q upon the second Fourier transformation, i.e. the upper half or even part of REAL' and the lower half or odd part of IMAGINARY' identifies these components with the terms resulting from the product $\hat{X}\hat{Y}$ or the transform operation associated with (X * Y). Thus the second transformation brings together the odd and even components of Q, i.e. the convolution of X with Y, in the real part of the output. The components which transform to become W are by-products of the entire operation but they are irrelevant. Figure 5 illustrates the

crosscorrelation process by a similar method, the difference being the initial inversion operation indicated. The real part of Z that is RE is inverted immediately after the first Fourier Transformation. Inversion means replacement of the independent variable $\hat{t}$ by $-\hat{t}$, which is the same as rotating $\hat{Z}$ by 180 degrees. Inversion is also equivalent to multiplying the odd symmetry component of a function by (-1). Since the odd symmetry component of RE is the imaginary part of the transform $\hat{Y}$, inversion here is equivalent to replacing $\hat{Y}$ by $\hat{Y}^*$. By definition this replaces the convolution function in the output, Q, by the crosscorrelation function.

Figure 6 illustrates another shortcut technique making use of implicit transforms. The task is to convolve two independent real functions X and Y each with a third real function T. T has the very convenient property that it is also evenly symmetric, so that, as a result $\hat{T}$ will be completely real. It is assumed that $\hat{T}$ has been pre-calculated and stored. As in the previous examples the complex function Z is formed from two real functions X and Y. After Fourier transformation, both RE and IM are multiplied by T. Now RE' and IM' are mutually exchanged. Such an exchange is one way of producing an inverse Fourier transformation except that the results of the second transformation are also exchanged, with the real part containing Y * T, and the imaginary part, X * T.

Figure 7 illustrates the deconvolution process carried out in exactly the same manner. RE and IM are each divided by T and the deconvolution functions result in the output. The symbol */ indicates deconvolution.

SHORTCUTS USING EXPLICIT TRANSFORMS

Figures 8 - 11 outline shortcut examples which assume explicit extraction of $\hat{X}$ and $\hat{Y}$ as an initial step. (Refer to figure 3) Once $\hat{X}$ and $\hat{Y}$ have been separated from $\hat{Z}$, they can be operated on directly to produce any Fourier process desired. If the results are grouped carefully to form a new $\hat{Z}$, successive Fourier transformation can result in two meaningful outputs instead of one.

Figure 8 demonstrates simultaneous convolution and crosscorrelation by working with the extracted Fourier components $\hat{X}e$, $\hat{X}o$, $\hat{Y}e$, and $\hat{Y}o$. The components grouped to form $\hat{Z}$ represent both $\hat{X}\hat{Y}$ and $\hat{X}\hat{Y}^*$. The components which travel to form Q upon successive transformation, including of course the (-1) are the terms resulting from $\hat{\hat{X}}\hat{\hat{Y}}$. Those which travel to form W are the terms resulting from $\hat{\hat{X}}\hat{\hat{Y}}^*$.

In Figure 9 the very same technique is used to produce both deconvolution and the deconvolution spread function. The deconvolution spread function W is defined in such a way that X * W = Q.

Figure 10 combines the previous examples producing both deconvolution and crosscorrelation at Q and W respectively.

Lastly, figure 11 presents the relatively simple example of producing two autocorrelation outputs; Q the autocorrelation function of X, and W the autocorrelation function of Y.

SUMMARY

What has been presented in this paper is a basic technique which allows the FFT Algorithms to be employed most efficiently. All of the techniques are based on the fact that the input data are completely real and the resulting complex Fourier transforms contain inherent symmetry. It is this inherent symmetry which allows the transforms of two real functions to be joined together as one transform operation and yet operated on as though they existed separately. Also, as we have shown, it is sometimes possible to perform two Fourier processing operations involving the same two functions in almost the same time and space. When the shortcuts are utilized in a machine program processing, time and memory requirements run 50 - 75% less than straightforward application of the FFT Algorithms; savings which can be quite significant when large volumes of data are to be processed and either computer processing time or memory space is limited.

REFERENCES

1. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol. 19, No. 90 (April 1965) pp. 297-301

2. T. G.Stockham, "High Speed Convolution and Correlation", AFIPS, Vol. 28, 1966 Sprint Joint Computer Conference, Spartan Books, Washington, 1966, pp. 229-233

3. W. M. Gentleman and G. Sande, "Fast Fourier Transforms - For Fun and Profit", 1966 Fall Joint Computer Conference, AFIPS, Vol. 29, Spartan Books, Washington, 1966, pp. 563-578

4. C. Bingham, M. D. Godfrey, and J. W. Tukey, "Modern Techniques of Power Spectrum Estimation", IEEE Transactions on Audio and Electroacoustics, Vol. AU-15, No. 2, June 1967, pp. 56-66

5. D. A. Swick, "Discrete Finite Fourier Transforms: A Tutorial Approach", Naval Research Laboratory Report 6557, June 1967 (Available as AD656577 from CFSTI, Dept. of Commerce, Washington, D. C.)

APPENDIX A - Demonstration of Symmetry Properties

The symmetry properties will now be demonstrated. The properties are:

A - The FDFT of a real even function is a real even function,

B - The FDFT of a real odd function is an imaginary odd function,

C - The FDFT of an imaginary even function is an imaginary even function,

D - The FDFT of an imaginary odd function is a real odd function.

An imaginary function is defined here as a function whose values are i times a real number.

The notation used is this: hatted variables denote variables in the transform (or frequency) domain, while unhatted variables denote space (or time) domain variables.

Let $X(t)$, $t = 0, 1, 2, \ldots, N-1$ be the sequence to be transformed, and let $X(t)$, $t = 0, 1, 2, \ldots, N-1$ be the Finite Discrete Fourier Transform (FDFT) of $X(t)$.

In order to demonstrate the symmetry properties of the finite discrete Fourier Transform, the following must first be proved:

1) the FDFT of a real function is hermitian symmetric, that is, if the transform is inverted, then its complex conjugate is obtained. Inversion here means rotating the function by 180 degrees.

2) The FDFT of an inverted function is itself inverted.

To demonstrate the hermitian symmetric property, replace $\hat{t}$ by $-\hat{t}$ in the defining equation of the FDFT, equation 1

$$\hat{X}(\hat{t}) \equiv \sum_{t=0}^{N-1} X(t) \exp(2\pi i t\hat{t}/N) \qquad \text{eq. 1}$$

$$\hat{X}(-\hat{t}) = \sum_{t=0}^{N-1} X(t) \exp(-2\pi i t\hat{t}/N) \qquad \text{eq. 2}$$

If $X(t)$ is real,

$$\hat{X}(-\hat{t}) = \left[\sum_{t-0}^{N-1} X(t) \exp(+2\pi i t\hat{t}/N)\right]^{*} \qquad \text{eq. 3}$$

$$\hat{X}(-\hat{t}) = [\hat{X}(\hat{t})]^{*} \qquad \text{eq. 4}$$

where * denotes the complex conjugate. Equation 4 defines a Hermitian symmetrical function.

The remaining symmetry properties are based on the following equation:

$$\hat{X}(-\hat{t}) = \text{FDFT}\,[X(-t)] \qquad \text{eq. 5}$$

i.e., if $X(t)$ is inverted, $\hat{X}(\hat{t})$ becomes inverted. This can be shown from the definition of the FDFT. Replace $\hat{t}$ by $-\hat{t}$ in equation 1.

$$\hat{X}(-\hat{t}) = \sum_{t=0}^{N-1} X(t) \exp(-2\pi i t\hat{t}/N) \qquad \text{eq. 6}$$

In order to place the right side of equation 6 in standard form

let $y = N-t$, so that $X(t) = X(N-y) = X(-y)$, and

$$\hat{X}(-\hat{t}) = \sum_{y=N}^{1} X(-y) \exp[2\pi i\hat{t}(N-y)/N] \qquad \text{eq. 7}$$

$$\hat{X}(-\hat{t}) = \sum_{y=N}^{1} X(-y) \exp[+2\pi i\hat{t}y/N] \exp[-2\pi i\hat{t}]$$

$$= \sum_{y=N}^{1} X(-y) \exp(2\pi i\hat{t}y/N) \qquad \text{eq. 8}$$

Since $\hat{t}$ is an integer, $\exp[-2\pi i t] = \exp[-2\pi i 0] = e^0 = 1$.

$$\hat{X}(-\hat{t}) = X(-N)\exp(2\pi i \hat{t}\, N/N) + \sum_{y-1}^{N-1} X(-y)\exp(2\pi i \hat{t} y/N) \qquad \text{eq. 9}$$

Since $\hat{t}$ is an integer,

$$\exp(2\pi i \hat{t} N/N) = \exp(2\pi i \hat{t}\, 0/N). \qquad \text{eq. 10}$$

Because X is periodic, $X(-N) = X(0)$. Then

$$X(-N)\exp(2\pi i \hat{t} N/N) = X(0)\exp(2\pi i \hat{t} 0/N) \qquad \text{eq. 11}$$

$$\hat{X}(-\hat{t}) = \sum_{y=0}^{N-1} X(-y)\exp(2\pi i t y/N) \qquad \text{eq. 12}$$

The right side of equation 12 is now in the form of equation 1.

Since y is a dummy index,

$$\hat{X}(-\hat{t}) = \text{FDFT}[X(-t)] \qquad \text{eq. 13}$$

To show that the FDFT of a real even function is real and even (Property A), use the previous two results. Let $X_e(t)$ be an even function, such that

$$\hat{X}_e(-\hat{t}) = X_e(t) \qquad \text{eq. 14}$$

Let $\hat{X}_e(\hat{t}) \equiv \text{FDFT}[X_e(t)]$ Then

$$\hat{X}_e(\hat{t}) = \text{FDFT}[X_e(t)] = \text{FDFT}[X_e(-t) = \hat{X}_e(-\hat{t}) = [\hat{X}_e(\hat{t})]^* \qquad \text{eq. 15}$$

$\hat{X}_e(\hat{t})$ is even

$\hat{X}_e(\hat{t})$ is real } Property A

which demonstrates the required result.

To show that the FDFT of a real odd function is imaginary and odd (Property B), let $X_0(t)$ be real and odd, such that

$$X_0(-t) = -X_0(-t) \qquad \text{eq. 16}$$

and let $\hat{X}_0(\hat{t})$ be its FDFT.

$$\hat{X}_0(\hat{t}) = \text{FDFT}\,[X_0(t)] = \text{FDFT}\,[-\hat{X}_0(-\hat{t})] = -\hat{X}_0(-\hat{t}) = -[\hat{X}_0(\hat{t})]^* \qquad \text{eq. 17}$$

(odd; imaginary)

which demonstrates the required property, property B.

The symmetry properties for imaginary functions (Properties C and D), are proven analogously to those for real functions. Let $X_e(t)$ and $X_0(t)$ be real even and real odd functions respectively. Then $iX_e(t)$ and $iX_0(t)$ are imaginary even and imaginary odd functions respectively. The FDFT is a linear operation. (See equation 1.) By the previously demonstrated properties,

$$\text{FDFT}\,[iX_e(t)] = i\ \text{FDFT}\,[X_e(t)] = i\hat{X}_e(\hat{t}) \qquad \text{eq. 18}$$

$$\text{FDFT}\,[iX_0(t)] = i\ \text{FDFT}\,[X_0(t)] = i\hat{X}_0(\hat{t}) = i[i\hat{X}'_0(\hat{t})] = -\hat{X}'_0(\hat{t}) \qquad \text{eq. 19}$$

where $\hat{X}_o(\hat{t})$ is a real and odd function, by Property B.

Since $\hat{X}_e(\hat{t})$ is a real even function (by Property A), property C is demonstrated.

Since $\hat{X}_o(\hat{t})$ is a real odd function (by Property B), property D is demonstrated.

APPENDIX B - Application of Symmetry Properties to Two-Channel Fourier Transformation

Let X(t) and Y(t) be two real functions whose transforms are desired. Express each function in terms of its symmetry components, defined as follows:

$$X_e(t) \equiv 1/2\,[X(t) + X(-t)] \qquad \text{eq. 20}$$

$$X_o(t) \equiv 1/2\,[X(t) - X(-t)] \qquad \text{eq. 21}$$

$Y_e(t)$ and $Y_o(t)$ are similarly defined. Note that these functions are even and odd in the conventional sense. The process shown in equations 20 and 21 is called symmetrization. These two equations may be added to get

$$X(t) = X_e(t) + X_o(t) \qquad \text{eq. 22}$$

$$y(t) = y_e(t) + y_o(t)$$

To effect a two-channel transformation, place the real function X in the real part of a new complex function Z, and place the real function y in the imaginary part of Z.

Let

$$Z(t) = X(t) + iY(t) \qquad \text{eq. 24}$$

Substitute equations 22, 23 in equation 24.

$$Z(t) = X_e + X_o + iY_e + iY_o \qquad \text{eq. 25}$$

Since the FDFT is a linear operator,

$$\hat{Z}(\hat{t}) = \hat{X}_e + \hat{X}_o + i\hat{Y}_e + i\hat{Y}_o \qquad \text{eq. 26}$$

Apply symmetry properties A, B, C, D respectively to the successive terms of equation 26.

$$\text{Real} \quad [\hat{Z}] = \hat{X}_e - \hat{y}_o \qquad \text{eq. 27}$$

$$\text{Imag} \quad [\hat{Z}] = \hat{y}_e + \hat{X}_o \qquad \text{eq. 28}$$

Since the even and odd parts of a function are separable by symmetrization (eq. 20, 21), the transforms $\hat{X}$ and $\hat{y}$ can be recovered

$$\hat{X}(\hat{t}) = \text{Even part of } Re\hat{Z} + i \text{ odd part of } Im\hat{Z} \qquad \text{eq. 29}$$

$$\hat{y}(\hat{t}) = \text{Even part of } Im\hat{Z} - i \text{ odd part of } Re\hat{Z} \qquad \text{eq. 30}$$

This separation may be carried out explicitly as above to get the separate spectra, or may be done implicitly to carry out various Fourier processing operations.

LIST OF FIGURES

BLANK PAGE

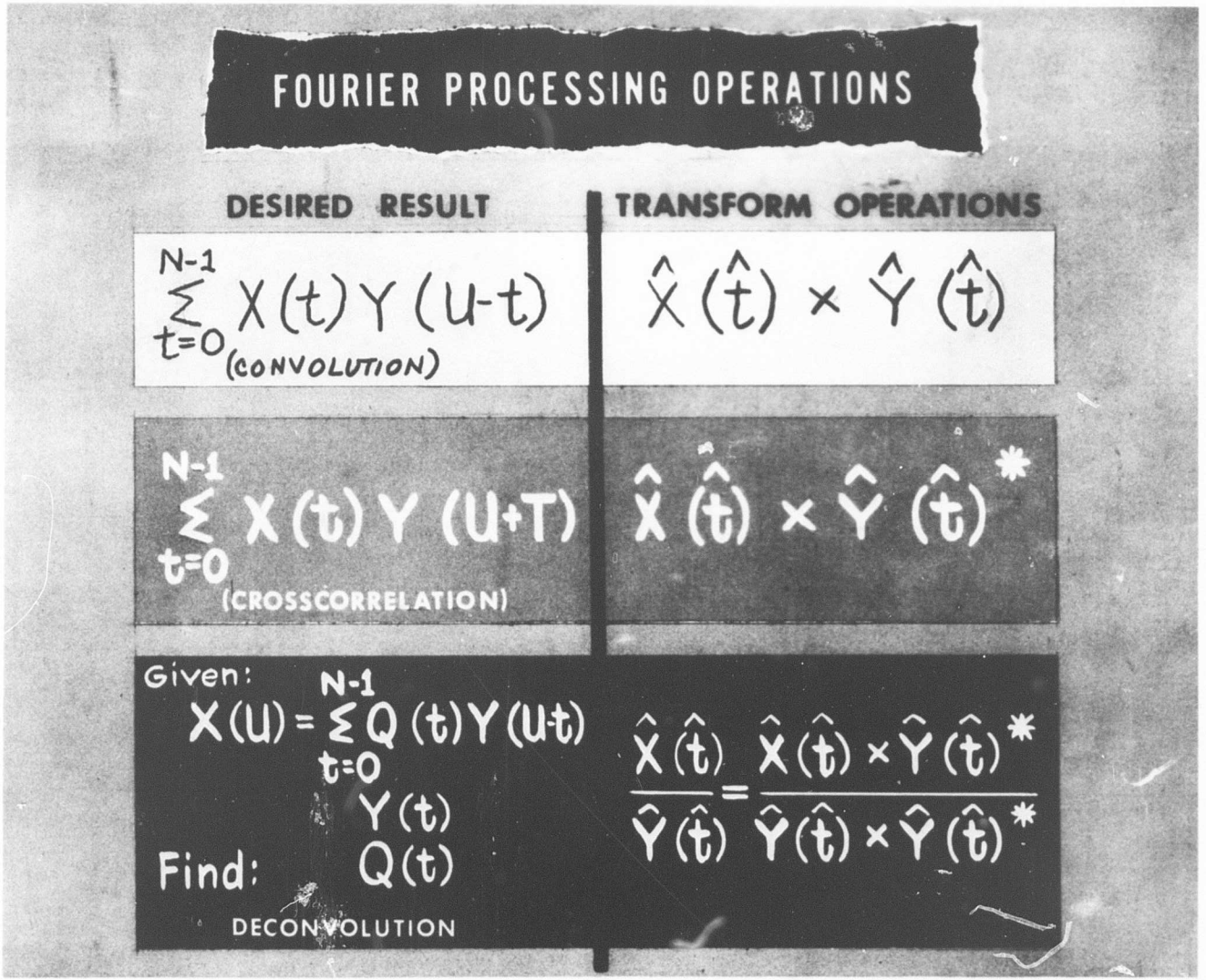

FOURIER PROCESSING OPERATIONS
DESIRED RESULT
TRANSFORM OPERATIONS
$\sum_{t=0}^{N-1} X(t)\,Y(u-t)$
(CONVOLUTION)
$\hat{X}(\hat{t}) \times \hat{Y}(\hat{t})$
$\sum_{t=0}^{N-1} X(t)\,Y(u+T)$
(CROSSCORRELATION)
$\hat{X}(\hat{t}) \times \hat{Y}(\hat{t})^*$
Given:
$X(u) = \sum_{t=0}^{N-1} Q(t)\,Y(u-t)$
Find: $\frac{Y(t)}{Q(t)}$
DECONVOLUTION
$\frac{\hat{X}(\hat{t})}{\hat{Y}(\hat{t})} = \frac{\hat{X}(\hat{t}) \times \hat{Y}(\hat{t})^*}{\hat{Y}(\hat{t}) \times \hat{Y}(\hat{t})^*}$

FIGURE 1

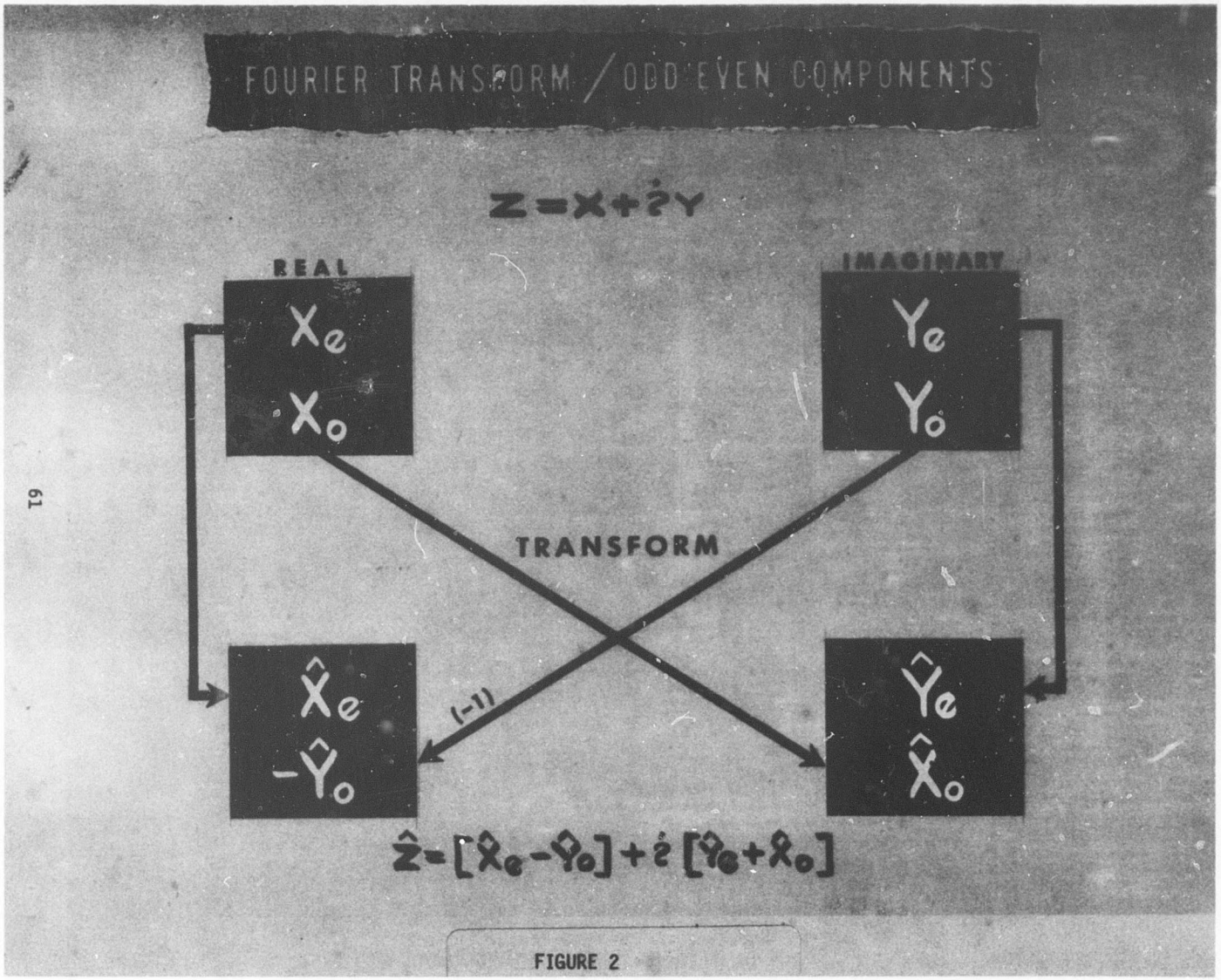

FOURIER TRANSFORM / ODD EVEN COMPONENTS
$Z = X + \hat{i}Y$
REAL
IMAGINARY
X_e
X_o
Y_e
Y_o
TRANSFORM
(-1)
$\hat{X}_e$
$-\hat{Y}_o$
$\hat{Y}_e$
$\hat{X}_o$
$\hat{Z} = [\hat{X}_e - \hat{Y}_o] + \hat{i}[\hat{Y}_e + \hat{X}_o]$

FIGURE 2

EXTRACTION OF FOURIER COMPONENTS

REAL

$\hat{X}_e$

$-\hat{Y}_o$

IMAGINARY

$\hat{Y}_e$

$+\hat{X}_o$

BY SYMMETRY:

$$\hat{X}_e(t) = \tfrac{1}{2}\{RE[\hat{Z}(\hat{t})] + RE[\hat{Z}(-\hat{t})]\}$$

$$\hat{Y}_o(t) = \tfrac{1}{2}\{RE[\hat{Z}(-\hat{t})] - RE[\hat{Z}(\hat{t})]\}$$

$$\hat{Y}_e(t) = \tfrac{1}{2}\{IM[\hat{Z}(\hat{t})] + IM[\hat{Z}(-\hat{t})]\}$$

$$\hat{X}_o(t) = \tfrac{1}{2}\{IM[\hat{Z}(\hat{t})] - IM[\hat{Z}(-\hat{t})]\}$$

FIGURE 3

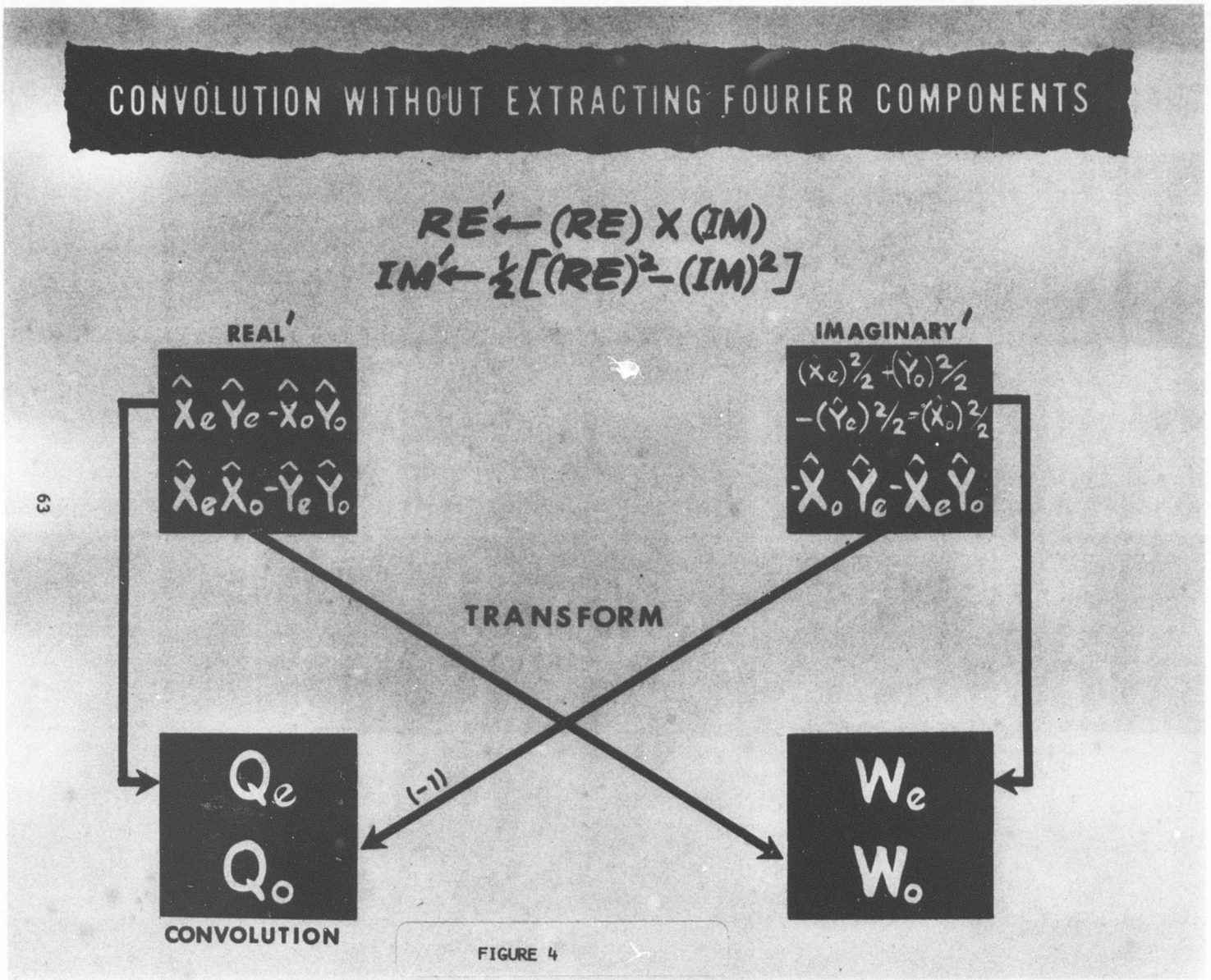

CONVOLUTION WITHOUT EXTRACTING FOURIER COMPONENTS
RE' ← (RE) X (IM)
IM' ← ½[(RE)² − (IM)²]
REAL'
IMAGINARY'
TRANSFORM
(-1)
CONVOLUTION

FIGURE 4

CROSSCORRELATION WITHOUT EXTRACTING FOURIER COMPONENTS

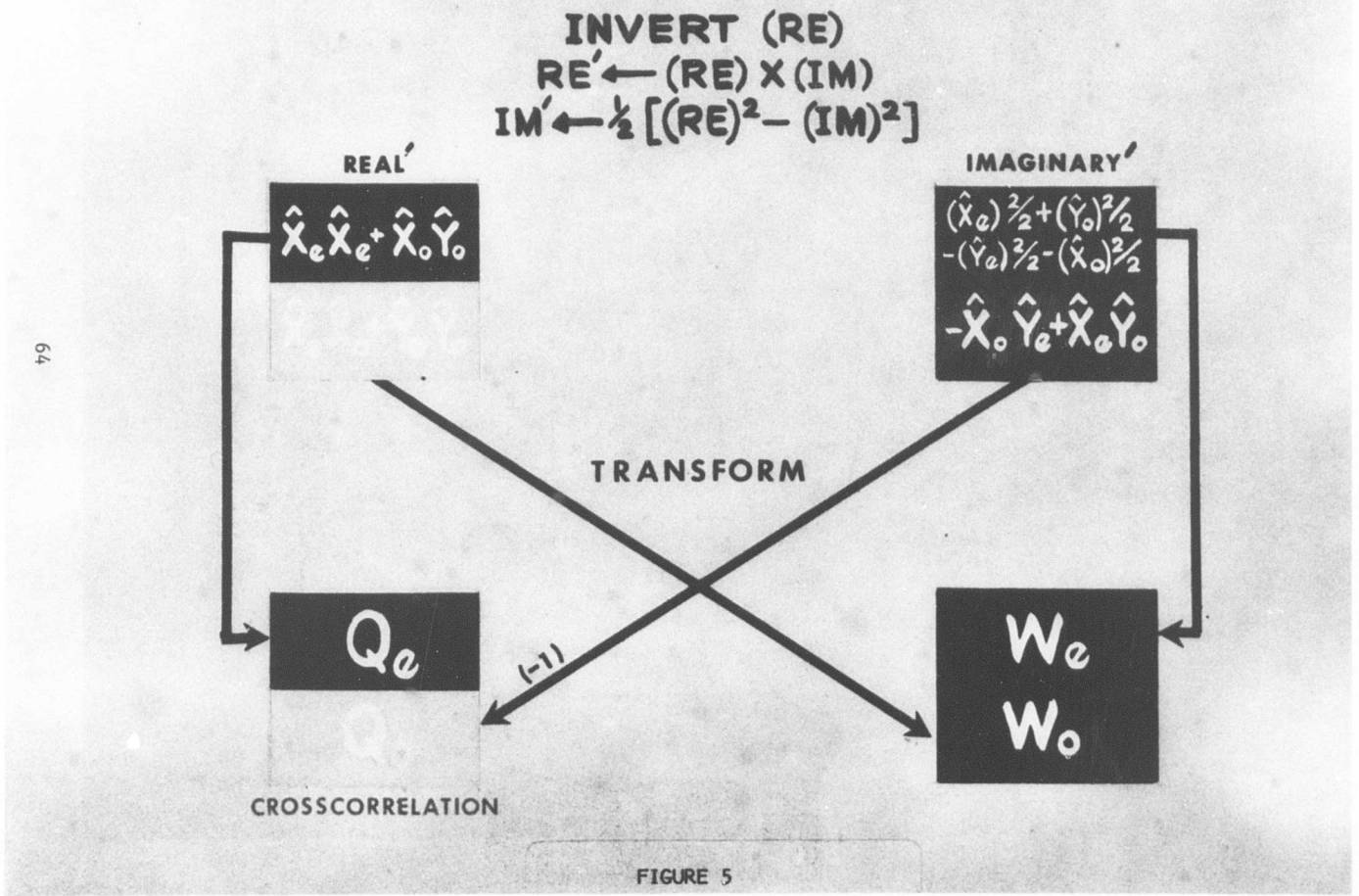


FIGURE 5

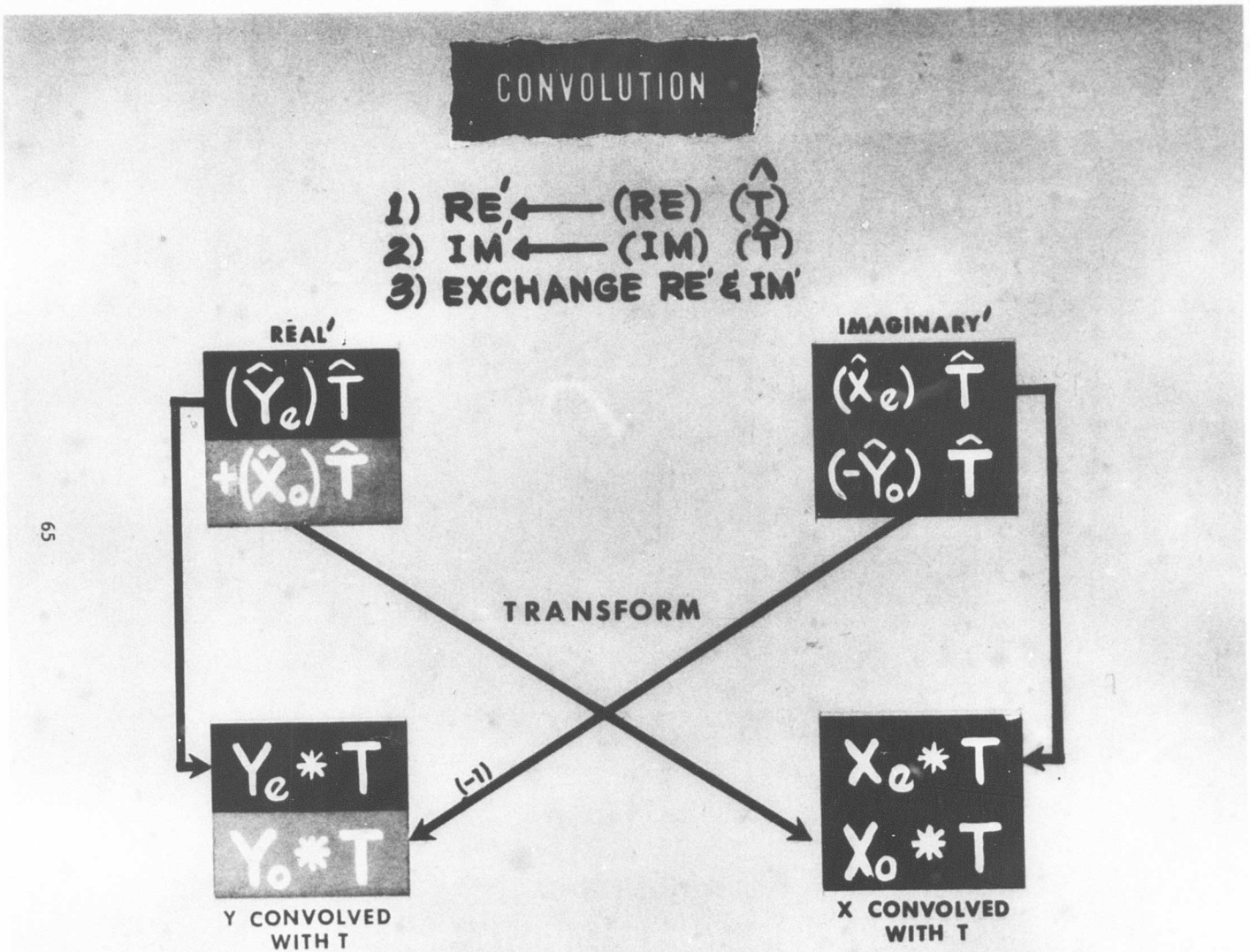

CONVOLUTION
1) RE' ← (RE) (T̂)
2) IM' ← (IM) (T̂)
3) EXCHANGE RE' & IM'
REAL'
IMAGINARY'
(Ŷe) T̂
+(X̂o) T̂
(X̂e) T̂
(-Ŷo) T̂
TRANSFORM
(-1)
Ye * T
Yo * T
Xe * T
Xo * T
Y CONVOLVED WITH T
X CONVOLVED WITH T

FIGURE 6

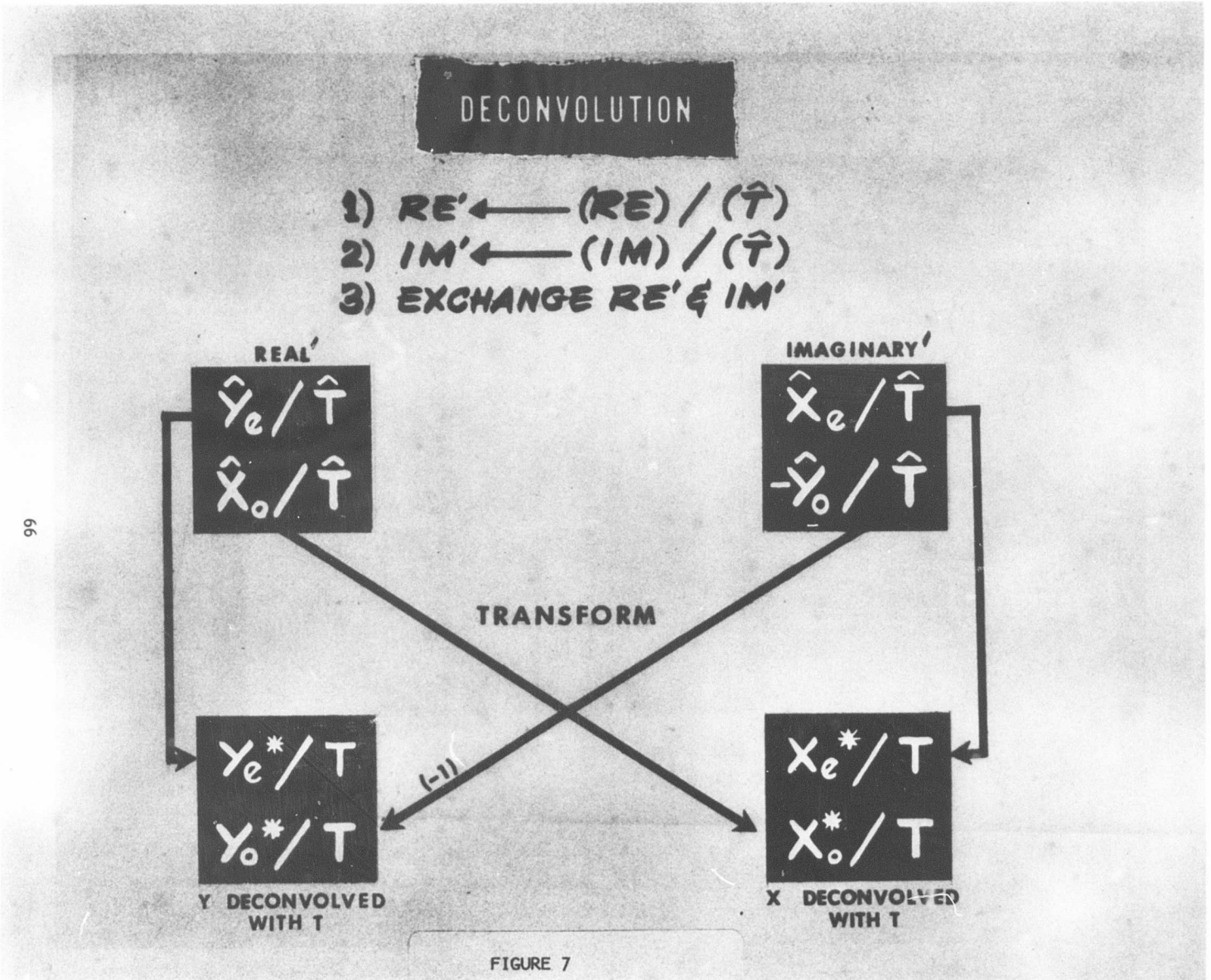

DECONVOLUTION
1) RE' ← (RE) / (T̂)
2) IM' ← (IM) / (T̂)
3) EXCHANGE RE' & IM'
REAL'
IMAGINARY'
Ŷe / T̂
X̂o / T̂
X̂e / T̂
-Ŷo / T̂
TRANSFORM
(-1)
Ye* / T
Yo* / T
Xe* / T
Xo* / T
Y DECONVOLVED WITH T
X DECONVOLVED WITH T

FIGURE 7

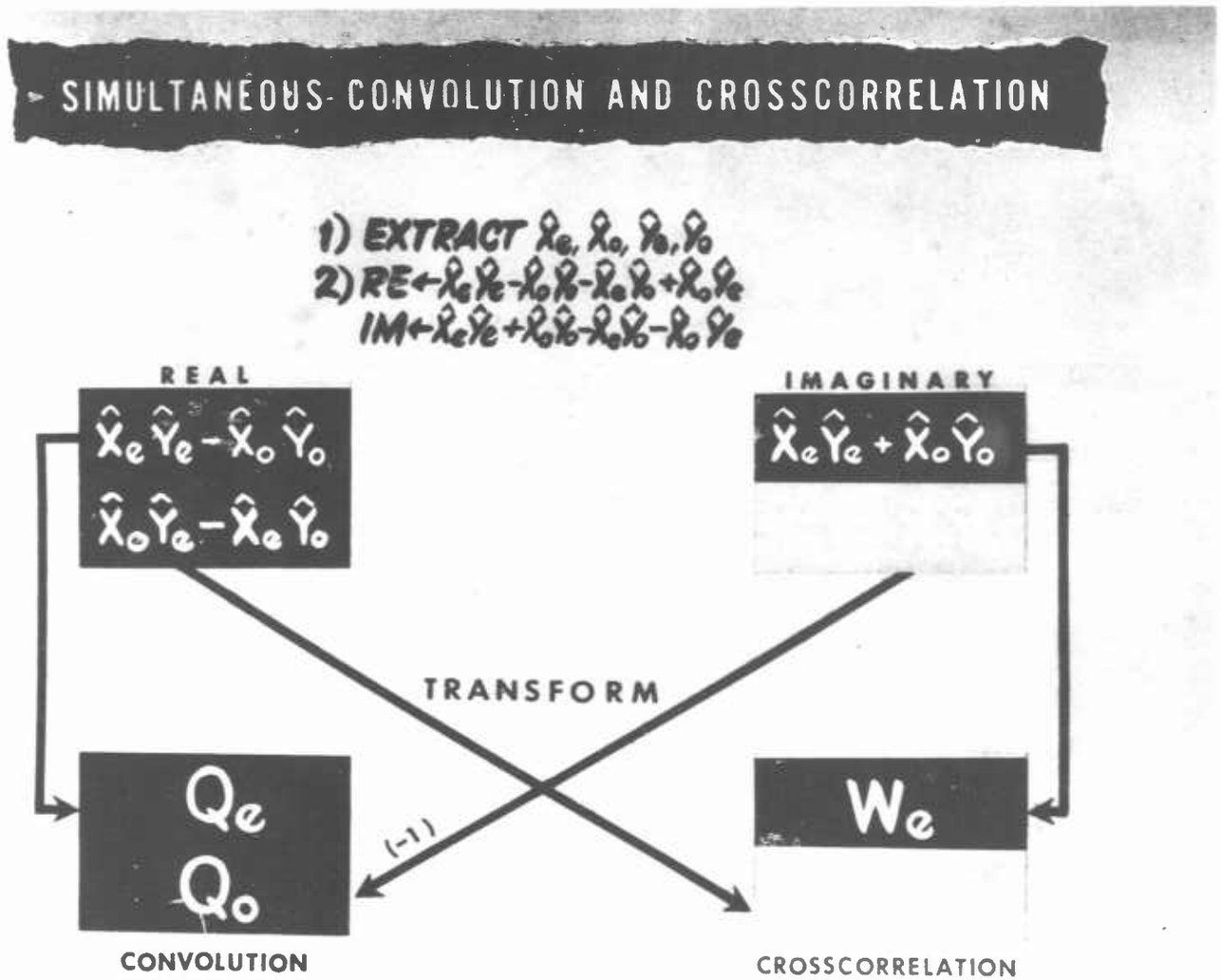

SIMULTANEOUS CONVOLUTION AND CROSSCORRELATION
1) EXTRACT $\hat{X}_e, \hat{X}_o, \hat{Y}_e, \hat{Y}_o$
2) RE ← $\hat{X}_e\hat{Y}_e - \hat{X}_o\hat{X}_o - \hat{X}_e\hat{X}_o + \hat{X}_o\hat{Y}_e$
IM ← $\hat{X}_e\hat{Y}_e + \hat{X}_o\hat{X}_o - \hat{X}_e\hat{X}_o - \hat{X}_o\hat{Y}_e$
REAL
$\hat{X}_e\hat{Y}_e - \hat{X}_o\hat{Y}_o$
$\hat{X}_o\hat{Y}_e - \hat{X}_e\hat{Y}_o$
IMAGINARY
$\hat{X}_e\hat{Y}_e + \hat{X}_o\hat{Y}_o$
TRANSFORM
Q_e
Q_o
(-1)
W_e
CONVOLUTION
CROSSCORRELATION

FIGURE 8

SIMULTANEOUS DECONVOLUTION & DECONVOLUTION SPREAD FUNCTION

1) EXTRACT $\hat{X}_e, \hat{X}_o, \hat{Y}_e, \hat{Y}_o$; $D \leftarrow \hat{Y}_e^2 + \hat{Y}_o^2$

2) $RE \leftarrow (\hat{X}_e \hat{Y}_e + \hat{X}_o \hat{Y}_o - \hat{Y}_o) \div D$

$IM \leftarrow (\hat{X}_e \hat{Y}_o - \hat{X}_o \hat{Y}_e + \hat{Y}_e) \div D$

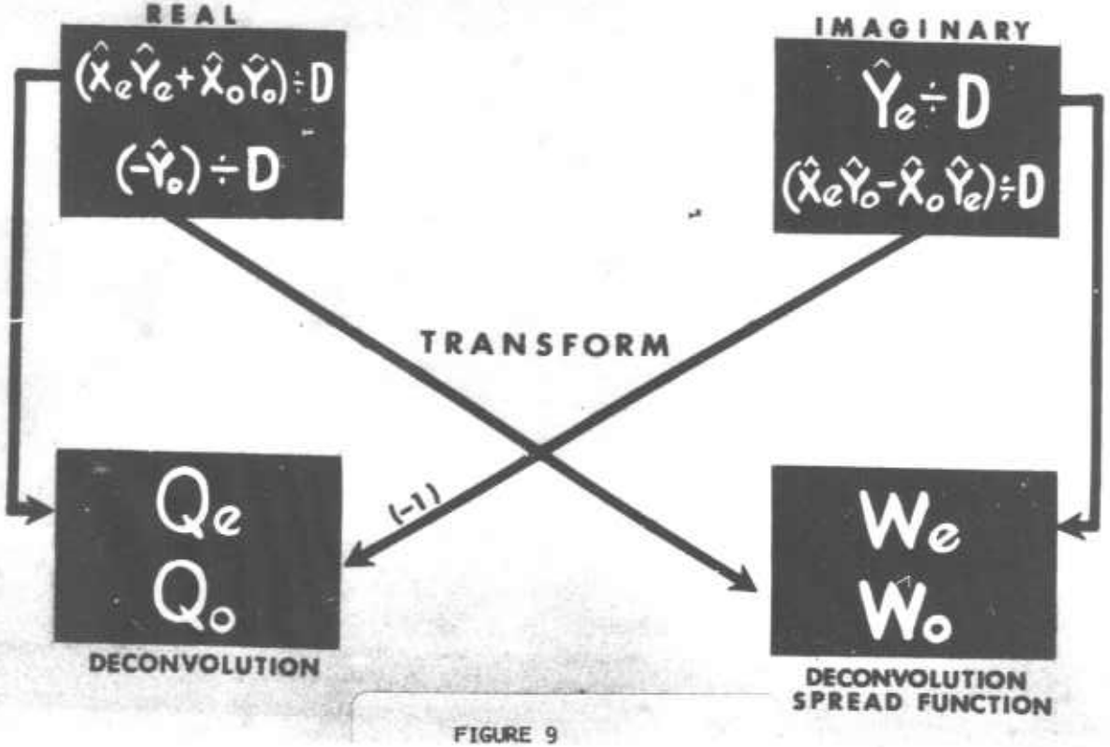


FIGURE 9

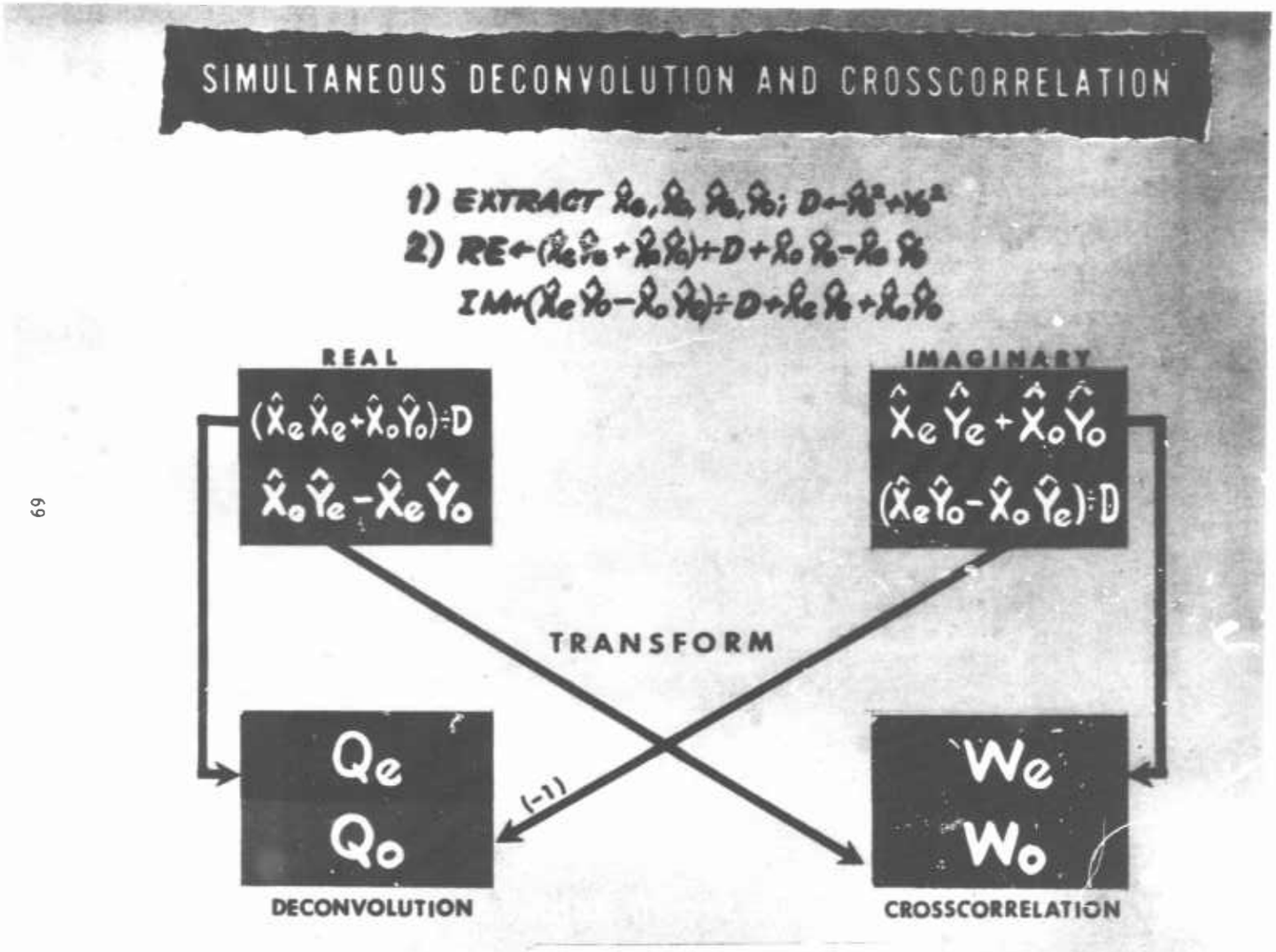

SIMULTANEOUS DECONVOLUTION AND CROSSCORRELATION
1) EXTRACT
2) RE
IM
REAL
IMAGINARY
TRANSFORM
(-1)
Qe
Qo
We
Wo
DECONVOLUTION
CROSSCORRELATION

FIGURE 10

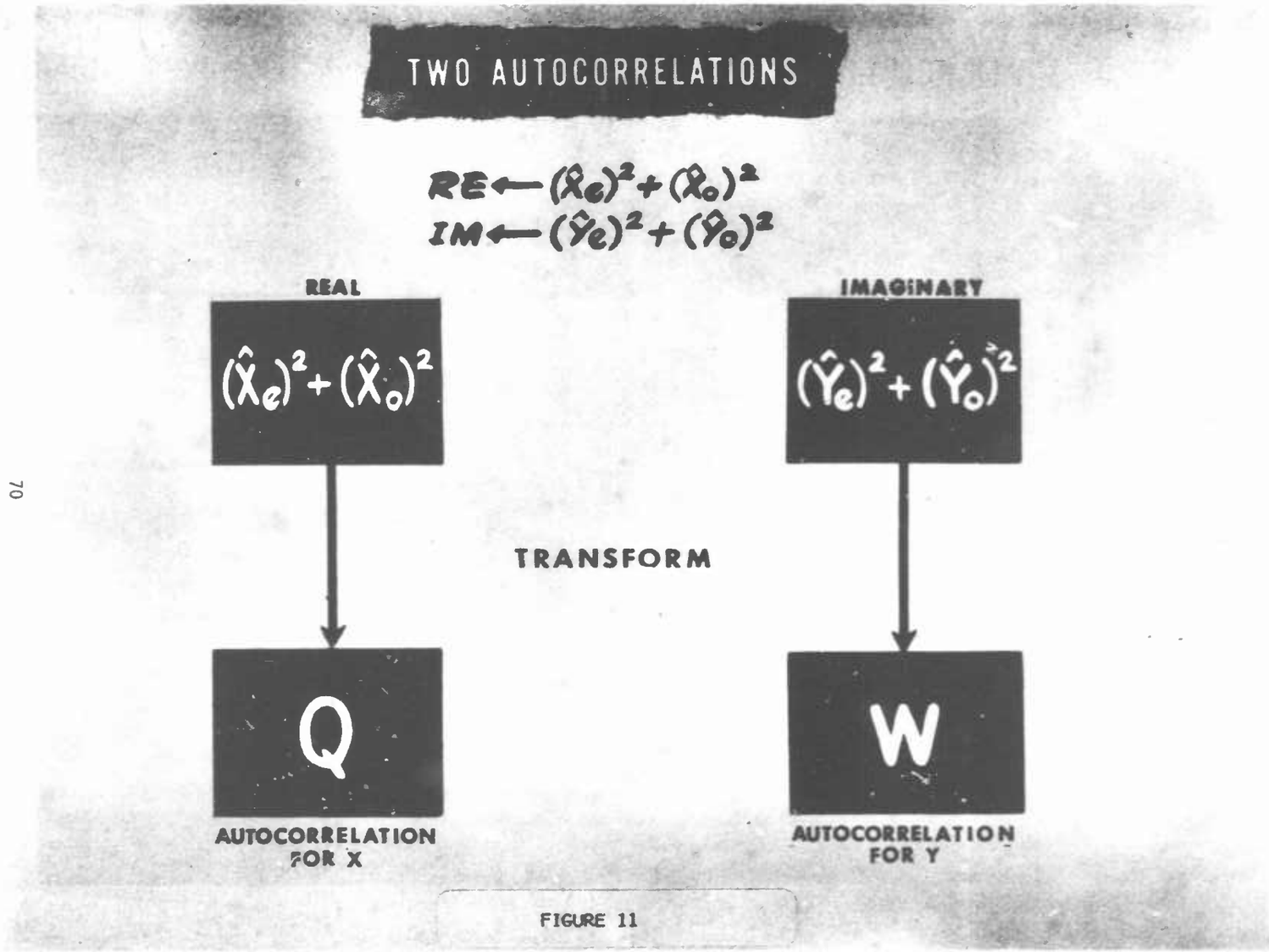

TWO AUTOCORRELATIONS
RE ← $(\hat{X}_e)^2 + (\hat{X}_o)^2$
IM ← $(\hat{Y}_e)^2 + (\hat{Y}_o)^2$
REAL
$(\hat{X}_e)^2 + (\hat{X}_o)^2$
IMAGINARY
$(\hat{Y}_e)^2 + (\hat{Y}_o)^2$
TRANSFORM
Q
W
AUTOCORRELATION FOR X
AUTOCORRELATION FOR Y

FIGURE 11

BLANK PAGE

COMPUTER PROGRAMMING FOR ACCURACY

J.M. Yohe
Mathematics Research Center, U.S. Army
The University of Wisconsin
Madison, Wisconsin

ABSTRACT. In most computations, it is tacitly assumed that the results produced by the program are "accurate enough". This assumption is not always valid. In this report, we discuss several possible sources of error in digital computation, and we list several steps which can be taken to guard against some types of error and determine an upper bound for the effects of other types. We illustrate these techniques with our experience in writing a program to locate zeros of the Riemann zeta function.

1. INTRODUCTION. In many cases, it is assumed that the results of a digital computation are right if they "look right"--whatever that means. This assumption is not always valid; indeed, sometimes it cannot be tolerated. For example, if a computation is done for the purpose of proving a mathematical theorem, the proof is no more reliable than the results of the computation. To put such a proof on reasonably solid footing, the computational results must be known to be correct with high probability. It would be best, of course, to prove the results correct with mathematical rigor; but the sources of possible failure are so numerous that such a proof is virtually impossible. We will therefore address ourselves to the problem of increasing the probability that the results are correct.

Unfortunately, the problems of detecting, eliminating, and/or controlling error are complex and ill-defined. Most of the textbooks on computer programming we have seen treat this subject lightly if at all; not because it is not important, but because any treatment of it is extremely difficult without using concrete examples.

In this report we will identify several general sources of error, and list specific steps which can be taken to either guard against error, or determine an upper bound on its effects. We will then illustrate these concepts with our experience in writing a program to locate roots of the Riemann zeta function. The relevant properties of this function are covered in [8] and aspects of the mathematical analysis, programming, and running of the programs are covered in [7]. The project was initiated by J. Barkley Rosser and Lowell Schoenfeld, and the computation was designed by Schoenfeld and the author. The computations were run on the CDC 3600 at the University of Wisconsin Computing Center during the period 1964-1968.

Sponsored by the Mathematics Research Center, United States Army, Madison, Wisconsin, under Contract No. DA-31-124-ARO-D-462.

The remainder of this paper was reproduced photographically from the author's manuscript.

We will limit ourselves to matters of producing computed results for correctly formulated problems. Of course, there is a possibility of error in the formulation and analysis of the problem, but this is not usually the concern of the programmer. In any case, many of these considerations have been treated in the literature; for example, in [4] there are exhaustive treatments of many problems, and Volume I of [4] contains an extensive bibliography. A general discussion may be found in [3]; [10] is another valuable reference.

No claim is made that this paper exhausts all possible sources of error, or that the methods suggested are best possible. However, it is hoped that it will prove valuable as a starting point for the planning of computations whose results must be accurate.

2. Sources of error. In this section, we give a condensed list of steps which might be taken to deal with various sources of error; in the remaining sections, we expand on these ideas and illustrate some of these steps with concrete exar ples.

A. Errors due to hardware limitations:

1. Determine the limitations of the hardware and the action of each instruction on the data.
2. Determine the type of error analysis to be used.
3. Plan the computation so as to minimize the effect of hardware limitations.
4. Make certain that any changes in hardware will be adequately publicized; check all such changes for possible effect on program.

B. Errors due to software limitations:

1. Determine the limitations of the software, including the accuracy of all subroutines to be used (including input/

output conversions); perform error analysis on all mathematical and input/output subroutines, both those used by the program and those used by the compiler/assembler.

2. Decide whether library subroutines should be used or whether special purpose routines should be written.
3. Make certain that any changes in software will be adequately publicized; check all such changes for possible effect on program.

C. Errors due to hardware failure:

1. Make sure the hardware operates as per specifications.
2. Determine what checking features are built into the hardware.
3. Determine what checks can be built into the program to help detect hardware failure.
4. Run the program on two different computers or run a second program to check thoroughly the results of the first program or run the same program twice and compare the results or design the program to effectively do the computation twice, using different methods.
5. Design the program to avoid or minimize use of hardware features that are either untested or are known to be troublesome.
6. If tapes are used, use them as little as possible and record in the lowest density practicable.

7. Design program to run efficiently in relatively short segments (say 1/2 hour or less) rather than requiring long unbroken runs.

D. Errors due to software failure:

1. Check that all software to be used performs correctly and to specifications.
2. Determine what checks are built into the software.
3. Check that software documentation is complete, concise, and accurate. Test all features of software for which documentation is cloudy, and document the results of the test.
4. If diagnostic capabilities of the compiler are not exhaustive, compile the program using another compiler for checking purposes.
5. Check the machine language program produced by the compiler.
6. Determine what checks can be built into the program to help detect software failure.

E. Errors due to program failure:

1. Flow chart the program.
2. Check the flow chart (both programmer and problem originator should do this).
3. Prepare a glossary of variable names used in the program.
4. Keep the flow charts and glossary up to date as programming progresses.

5. Desk-check program (a second programmer should do this).
6. Check the program against another program, a hand computed case, or against published tables. Explain completely all discrepancies.
7. Check that all branches of the program work properly and interact properly.
8. Provide for expansion of the program and/or its task without undue reprogramming. (for example, use variables for limits of loops and make I/O unit designations general).

F. Errors due to faulty operation:

1. Design the program for ease of operation and minimal action by the computer operator.
2. Determine what checks the program can make on input data.
3. Provide for print-out of all input data for visual checking whenever possible.
4. Determine scrupulously the effects of program interrupts by the operator, or of other steps that the operator may initiate during the running of the program.

G. Errors due to inadequate planning:

1. Determine in detail the present and future objectives of the program.
2. Determine what input and output is required for all purposes, and the formats to be used for it.
3. Determine the names to be used for variables in the program.

4. Determine what checks will need to be made, how they will be made, and decide how the computation can best be designed to facilitate checking.
5. Make adequate provisions for restarts.
6. Take into consideration all points listed in the previous six categories.

3. Errors due to hardware limitations. The most prevalent error of this type is the so called "round-off error". It cannot be eliminated, but its effect can be determined since it is systematic and completely predictible, given that the action of the hardware on data is known for the various operations. (The computation can usually be designed so as to reduce the effect of round-off error, but we will not attempt to discuss this here). There is a large body of work dealing with the analysis and control of round-off error. We will mention two such methods.

One popular method of error estimation is interval arithmetic, which was developed by R. E. Moore; it is detailed in [2]. A general discussion of interval arithmetic, some of its applications, and techniques for programming it can be found in [5], and an implementation of interval arithmetic for the CDC 1604 and the CDC 3600 can be found in [6].

Interval arithmetic has the advantage of being relatively easy to use. Its principal disadvantages are that it is perhaps somewhat slower than other methods, and the facility for performing interval arithmetic is not, to our knowledge, incorporated in the hardware of any popular computer. Therefore, interval arithemetic must be performed by a program. Such programs are not generally available for most computers and, even if they are available, they are subject to many of the same

sources of error as any other program.

Another method of determining bounds on this type of error is the a priori error analysis introduced for floating point arithmetic in 1960 by Wilkinson [9]. In Chapter IX of [11], Lowell Schoenfeld discusses this method, compares a priori analysis with interval arithmetic, and gives specific values of Wilkinson's parameters for error analysis of CDC 3600 computations.

In the zeta function programs, we elected to use the a priori analysis. The primary reason for this choice was speed of computation. The analysis was checked carefully, and then the error bounds obtained were multiplied by a factor of 2.1 for added safety.

An a priori error analysis is properly part of the planning of a computation. However, for this type of analysis, the action of each machine instruction must be completely known, and the precise order in which the instructions are executed must also be known. Consequently, the planning and the programming must be interactive in this case. The program affects the error analysis, and the error analysis in turn may reveal places in the program where modification would yield a more accurate computation.

Such error analysis involves a great deal of tedious calculation. Although this may seem to be an unnecessary amount of work, it has values other than just bounding the error in the final computation. For example, these error bounds alerted us to an error in the compiler which would probably never have been noticed otherwise; see [7].

We should not leave this subject without mentioning one other point. On some computers, it is possible to alter the rounding algorithm; programs

may then be run twice, using a different rounding algorithm each time. While this method may well be of value in deciding whether the answers "look right", it does not in general yield rigorous bounds on this type of error.

4. Errors due to software limitations. As the hardware has certain limitations in its performance, so does the software. In this section, we assume that the software is functioning as designed, and that therefore errors of this type will be systematic and completely predictable once the action of the software is known.

Software limitations include errors in mathematical functions, errors in binary-to-decimal and decimal-to-binary conversion (both in compilers and assemblers and in the library subroutines used by the program at object time) , etc.

If listings are available and not too many routines are to be used, one may elect to go ahead with an a priori analysis. This is tedious and time consuming work, and one always runs the risk that the routines will be changed after the analysis is completed. In the zeta function programs we found during debugging that the results of two consecutive runs differed in places where presumably no changes had been made. Investigation showed that several decimal constants had been converted to binary differently in the two runs. The reason was, we found, that the conversion routine used by the compiler had been changed between the two runs, and the two routines gave slightly different results. We had not been informed of this change.

An a priori analysis of input-output routines may be extremely difficult due to the complexity of the routines. For the mathematical routines, the

a priori analysis involves analysis of the error due to mathematical approximation as well as error due to hardware limitations. The job is, indeed, a monumental one.

Another alternative is to write home-made routines for the jobs which need to be done. For example, in the zeta function program, we wrote a special high-precision conversion routine to convert critical constants to binary. The binary numbers were then given to the compiler, which merely copied them; hence no error was introduced by the compiler. Similarly, we wrote our own programs for logarithm, square root, and cosine.

The reader may wonder why all of this is necessary. After all, the manufacturer does provide error bounds, at least on the mathematical routines; why not use those? The answer is that the bounds customarily provided by the manufacturer are either bounds on the mathematical approximation or bounds determined empirically; we know of no case where the bounds given are asserted to be mathematically rigorous.

5. Errors due to hardware failure. Up to this point, we have considered only predictable and systematic errors. We now embark upon a consideration of errors which are less frequent, totally unpredictable, and almost totally outside the control of the programmer. Errors of the type to be considered in this section include main frame failure, failure of disks, drums, card readers, tape units, etc. The list is endless, or seemingly so. We will also include here errors due to failure of the magnetic tapes themselves.

Such errors are difficult to find and difficult to guard against. All reasonable checks which can be built into a program should be (such checks are valuable for catching other types of errors as well).

For example, it is reasonable to test that the sign of a computed result is correct, or that the magnitude of the computed result is within certain bounds, if such infcrmation is known. The nature of such checks for a particular problem will depend on the analysis of the problem in question, and no general rules can be given.

One of the surest methods of catching errors due to hardware failure is to run the entire computation twice -- preferably using different programs and different machines. This, however, may not be feasible. If the computation can be run twice on the same machine using totally different programs, this would be nearly as good. Either of these two alternatives is also an effective check for other types of errors. A third choice is to run the computation twice on the same machine using slightly different programs. The answers obtained by the two runs should be compared and all discrepancies explained.

For the zeta function program, we effectively ran the computation twice, using slightly different programs. The first program computed certain values of the independent variable tau, (whose accuracy was not critical) and a certain function f of tau. The values of tau were chosen so that consecutive values of f(tau) had opposite signs and were larger in magnitude than the a priori error bounds. The first program then wrote the tau values and the sign of f(tau) on magnetic tape. The second program read the tape, recomputed f(tau), and checked to make sure that the sign of f(tau) read from tape agreed with the sign just computed. The program also made several other checks.

The hardware was not entirely error free, although it did prove to be remarkably reliable (see [7]). On one occasion, we had trouble due to a bit

dropping in one of the CPU registers. Fortunately, this manifested itself by causing either an attempted write on a nonexistent tape or a format length error on printer output. In either case, the operating system merely aborted the program.

There were two other means of checking the actual computation on a spot-check basis. The first of these involved the use of a totally different program to perform the same computations. This program was applied to selected points from each run of the computation, and the results compared against the results obtained by the main program. The second method of checking involved the use of ten variations of the method of computing f(tau) (see [7]). Periodically, all ten methods were applied to one value of the independent variable and the results were compared. The results of each pair of methods were required to agree to within the sum of the a priori error bounds for the two methods.

The tapes and tape units gave us trouble practically from the beginning of the computation. The first program was designed so that, on a restart, it would first copy the tape from the previous run onto a fresh tape, and then add the results of the current computations to the end of the new tape. In this manner, the previous tape could be file protected, and the risk of damage or destruction was minimized. We rotated among three and sometimes four tapes, and this gave us a further measure of protection in the event that a run was aborted. This method seemed to work as well as could be expected with tapes; and since the hardware is designed for read-after-write and the software we used implements this check, we had comparatively few parity errors on the tapes we wrote. Those tapes which did seem to have bad spots on them when

reading was attempted could almost invariably be read by mounting them on the physical unit on which they were originally written. We also used nearly new tapes for the output.

As the second program progressed, tape failures became more frequent. We found that by switching from the previous recording density of 800 bpi to 556 bpi the troubles were alleviated to some extent; the remainder of the computation was recorded on tape at 556 bpi. We would have gone to 200 bpi, except that the number of reels of tape required would have been prohibitive.

Whenever two reels of tape were supposed to be identical, they were compared to make sure that they were identical. If they were not, any discrepancies were checked to determine which tape, if either, was correct. We always saw to it that we had duplicate copies of all important reels of tape; this saved having to rerun portions of the computation to recreate tape that could not be read.

Whenever possible, the program should be designed to run in relatively short segments rather than requiring long, unbroken runs. This will reduce the probability of hardware failure during any one run, and will also reduce the computer time lost if such an error does occur. The zeta function program was not designed this way, and there was an inordinate amount of computer time lost due to hardware (and software) failures.

6. Errors due to software failure. Software failure is not as infrequent as might be hoped, especially if the computer or the software is new. Like any other program, software must be debugged; yet most of us tend to assume that, by the time software is released, it is relatively bug-free. And, in fact, the

obvious bugs are generally out. That leaves the subtle ones to plague us. Dealing with these bugs requires all of the techniques mentioned thus far, together with some to be mentioned later.

Certain experiences with the zeta programs might be instructive. Other such experiences are noted in [7].

Various errors and inadequacies were found in the compiler. For example, the compiler converted a double precision constant in an IF statement to zero instead of the correct value. This manifested itself by branching the wrong way in the IF statement; print-outs from the program showed that this had happened. The error was identified by examining the machine language listing of the compiled program.

In the Drum Scope operating system we encountered one of the most serious bugs found in software during the entire project. Shortly after the Drum Scope operating system replaced the Tape Scope system, one of the checks built into the second program began failing. This would only occur once in a run, since the program was designed to stop immediately in this event. But on rerunning, the check would not fail. This was explained only after it was discovered that the monitor system failed to restore an internal register after the operator interrupted to type in a message. (This register was ordinarily used very little, but our program made heavy and consistent use of it throughout the computation.) It was then decided to rerun the entire computation and compare the results of the two runs; as it turned out, only one erroneous value had gone unnoticed (although another had been found by visual inspection of the output).

Another type of inadequacy which can occur in software is the "sin of

omission". For example, a FORTRAN compiler may fail to inform the programmer of some irregularity--such as a variable which is defined but never used. This will undoubtedly affect the results of the computation if, for example, the variable name was misspelled when it was intended to be used (or when defined). The compiler might then merely reserve a cell for the one spelling of the variable name, setting its contents, say, to zero; the programmer, not having been informed, may not realize that this has happened. For reasons of this sort, it is well to have the program compiled using the best diagnostic compiler available, even if this means that the program cannot make use of the more powerful language available on the object computer. When the program compiles successfully under the diagnostic compiler, it can then be recompiled for use on the computer best suited to do the computation. The use of a good diagnostic compiler will usually also result in a reduction of both man-hours and elapsed time required to get a program running.

Operating systems can have similar faults. For example, in the zeta function computation, a change in logical unit numbers for the purpose of beginning a new reel of tape was handled incorrectly by the program. This, of course, was not the fault of the operating system. However, one result was that the operating system was asked to write on a tape which had not been declared by the control cards. The monitor merely assigned a scratch tape, with no indication that this had been done; wrote on it, then released it at the end of the run--with the result that the information was lost. Had the monitor informed us of this assignment, no harm would have been done due to the rotation scheme of tapes; as it was, this programming blunder cost us nearly five hours of computer time.

7. Errors due to program failure. In this category we place all errors in preparing the problem for computation. These include errors in logic, flow charting, coding, transcription, keypunching, etc. -- the sorts of errors usually referred to as "bugs". Chapter 13 of [1] gives a general discussion of the problem of program debugging or checkout and lists some specific methods which can be used to determine the source of error once it has been determined that an error exists. And, of course, most programmers have developed their own methods of debugging through experience. The problem is in determining whether an error exists in the program.

A program should be flow charted before coding begins. All too often this step is skipped because "the problem is so simple that a flow chart is not necessary". But, particularly if the programming is being done for someone else, a flow chart is of great value. For example, the person originating the problem can usually check the flow chart (for this reason, it is a good idea to adopt some standard for flow charting symbols within the organization; see appendix for a suggested standard); if this is done, he may uncover errors due to lack of communication between himself and the programmer as well as errors in logic or design of the computation. The programmer should also prepare a glossary of variable names used.

Once programming has begun, it is rare indeed that the flow chart is followed exactly. There are usually places in the program where more efficient or more effective ways of doing the job occur to the programmer as he writes the program. This is to be expected; however, the flow chart should be updated and rechecked as the programming progresses.

When the program is complete, it should be desk-checked by another programmer (it is taken for granted that the programmer checks his own work)

before it is tested on the computer. However, it is usually a good idea for both the programmer and the checker to work from a listing of the card deck for the final pre-computer check; this also tends to reduce the number of key-punching errors which actually get to the computer. The initial number of key-punch errors will be reduced if the programmer writes the program neatly on the coding sheets and uses standard graphic symbols for characters which may be confused -- for example, O and 0 , I and 1 .

In particularly critical cases, the program should be written in two different languages or for two different computers and the results of test runs compared before debugging is considered complete. In the zeta function programs, critical subroutines were written for both the CDC 1604 and the CDC 3600, checked out on both machines, and then results of test runs were compared. In addition to the expected number of program bugs uncovered by this procedure, we found that the double precision arithmetic program used by 1604 FORTRAN was in error (it returned a value of zero for a-b if $|a-b|$ was less than $a \times 2^{-36}$), and that the floating divide instruction on the 1604 did not round correctly. Although both of these discoveries involved the machine we were <u>not</u> going to use for the final computation, this can only be regarded as fortuitous.

Of course, the results of the computation should be compared against a known check case, and this check case should encompass as many of the features of the program as possible. This case might be a hand-computed special case, a case computed by an existing program which is known to be correct, or values given in published tables. All discrepancies should be completely explained. (In one case we found an erroneous value in one of the tables we were using for checking purposes).

All possible branches of the program should be checked out, by fake

data if necessary. This not only assures that each branch works as it should, but also that the different branches interact properly.

Finally, when possible, the program should be checked by comparing the results obtained by another program using a totally different method (and, if possible, written by another programmer without collaboration.)

A word should be said about the choice of programming languages. In some cases, machine language offers such benefits that it virtually has to be used. (see, for example, [12]). In such cases, it should be used; however, errors are much easier to make in programming in machine language, and they are, as a rule, much harder to find. For the most part, one of the problem oriented languages should be used; preferably the one with the best diagnostic capabilities available. But here, too, caution is necessary; as we have noted, these compilers themselves may contain errors which might result in an erroneous program being compiled.

In the zeta function program, the necessity of obtaining a degree of efficiency impossible with FORTRAN, together with the necessity of complete control over operations used and operation sequences (due to the a priori error analysis) dictated the use of machine language for much of the programming. These portions of the program were also written in FORTRAN, and the two versions were compared. We used FORTRAN for the remainder of the program, but the listings of the machine language created by the compiler were carefully checked to make sure that the compiler indeed created a program which would do exactly what was intended.

8. Errors due to faulty operation. These errors include computer operator errors and errors in setting up the program for a run, preparing data, etc.

The best way of guarding against operator errors is to make the program as easy as possible to operate, and minimize the necessity for operator intervention. The program should check any tapes provided as input to make sure that the correct tape is being provided. In addition, all input tapes should be file protected. This will mean that the program cannot write on the input tape; if it is necessary to have a tape containing both the input and the output from the run, the input tape should be copied onto the output tape.

As far as errors in data preparation and program set-up are concerned, the program should be designed for ease of set-up and data preparation. For example, input might be free-format, and each quantity might be identified on the data card by its name. The program could then read the name, locate the parameter in the program, then read the value and store it. The program should also make thorough checks on the consistency of the input data, and print out the input data for later visual inspection.

9. Errors due to inadequate planning. Errors of this sort should be caught in the programming or, if they manifest themselves during the running of a program, they should be caught by the program. For example, if too little storage space is allotted for an array, the program should catch this in its check to make sure that the array length is not exceeded. (Note that such a check is not compiled automatically by most FORTRAN compilers.) Again, the tape-switching error described in Section 6 was definitely a planning error, and should never have reached the point of affecting the computation --but it did.

Several things can be done to help here. First, the planning of a

program should include provisions for the program to be expanded (or, more properly, for its task to be expanded) beyond the wildest dreams of the programmer and the project originator. For example, if the program is to use tape as an output medium, take it for granted that it will use an arbitrarily large number of reels, even though "it can't possibly fill up more than half a reel". Then plan accordingly.

One point which seems minor, but can make life easier for all concerned, is this: All variable names used in the program should agree as nearly as possible with the names used for them in the planning stages. This may mean that some variable names will not coincide with their usual type (integer or floating point) if FORTRAN is used, and extra declarations will be necessary; however, that price is small compared to the cost of confusion arising when there are a large number of variables, each of which has two names. In the zeta function programs, matters were arranged so that nearly all variables could be changed by supplying the program with a card giving the name of the variable and the value desired. The idea was good, and it worked well in practice, except that most variables had both "internal" and "external" names, and confusion was all too common. In fact, we later needed the capability of changing variables which had not originally been designed for changing and we accomplished this by specifying the location in the parameter array to be changed. This gave us yet a third name for most variables, and the confusion was somewhat disconcerting.

In the planning, adequate provision should be made for restarting the computation and for recovery of information lost due to hardware or software failure. The design of the zeta function program left something to be desired

in this regard. In the interest of minimizing the number of tapes used, we wrote the restart information on the output tape. The tape format should have been designed so that restart information could be interspersed with "good" output, and restart information should have been provided frequently. Unfortunately, this was not done; the result was that the only restart points were at the ends of successful runs. Consequently, a failure near the end of the run either caused the loss of the entire run, or necessitated tedious (and error-prone) preparation of restart information by hand.

Finally, the planning of the computation should include provisions for all of the contingencies discussed in the previous sections.

10. Conclusion. After all of these precautions have been taken and after all of this tedious and laborious detail has been attended to, can we be certain that the results of our computation will be accurate? No. At least, not 100% sure; there is always room for human error in human endeavors. But we have acknowledged the many possible sources of error, and we have approached the computation in a systematic and careful manner, with the specific intention of eliminating, minimizing, or controlling and completely determining the effects of these errors. We have not merely ignored the problem in the hope it will not occur; we have attempted to solve it. In that sense, we can place a good measure of confidence in the results, and the element of doubt that must certainly remain is intelligent doubt. In any case, we are certain that we have done the best we can.

REFERENCES

[1] McCracken, D. D. Digital Computer Programming. John Wiley & Sons, Inc., New York, 1965.

[2] Moore, R. E. Interval Analysis. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1966.

[3] Noble, Ben. Rounding error, ill-conditioning and instability. Proceedings of the 1967 Army Numerical Analysis Conference, ARO-D Report 67-3 (1967), 209-227.

[4] Rall, L. B., Editor. Error in Digital Computation, Vol. I and II. John Wiley & Sons Inc., New York, 1965.

[5] Reiter, A. Programming interval arithmetic and applications. Proceedings of the 1967 Army Numerical analysis Conference, ARO-D Report 67-3 (1967), 87-98.

[6] _____. Interval arithmetic package (INTERVAL) for the CDC 1604 and CDC 3600. Mathematics Research Center Technical Summary Report #794, January, 1968.

[7] Rosser, J. B., Lowell Schoenfeld, and J. M. Yohe. Rigorous computation and the zeros of the Riemann zeta function. Proceedings of the IFIP Congress 68, to appear.

[8] Titschmarsh, E. C. The Theory of the Riemann Zeta Function. The Clarendon Press, Oxford, 1951.

[9] Wilkinson, J. H. Error analysis of floating point computation. Numerische Math. 2 (1960), 319-340.

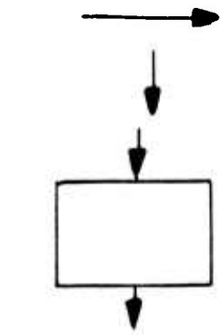
[10] Wilkinson, J. H. Rounding Errors in Algebraic Processes. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1963.

[11] Yohe, J. M. Machine language programming for the CDC 3600. Mathematics Research Center Technical Summary Report #721, August, 1967.

[12] _____. Machine language programming -- how and why. Proceedings of the 1967 Army Numerical Analysis Conference, ARO-D Report 67-3 (1967), 3-9.

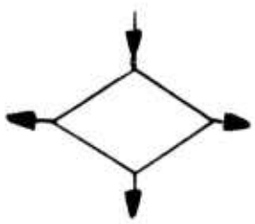
APPENDIX

SUGGESTED STANDARD FOR FLOW CHARTING SYMBOLS

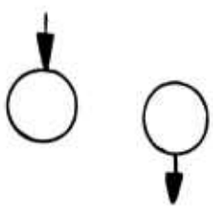


DIRECTION OF FLOW

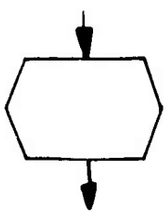
PROCESSING FUNCTION



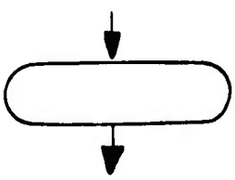
DECISION



CONNECTOR



PROGRAM MODIFICATION FUNCTION



INPUT-OUTPUT

BLANK PAGE

COMPUTATION OF THE DYNAMIC RESPONSE OF A MEMBRANE

Aivars Celmiņš
Applied Mathematics Division
U.S. Army Ballistic Research Laboratories
Aberdeen Proving Ground, Maryland

ABSTRACT

Typical deformations of membranes in dynamic response problems are large and, therefore, a linearization of the problem by assuming "small deflections" is not possible. For the same reasons the stress-strain laws cannot be assumed linear. The problem to be solved is then a non-linear totally hyperbolic second order system of partial differential equations. In a specialized case considered here the system consists of two equations for two unknown functions.

For the solution of equation systems of the type mentioned (i.e., $u_{tt} = f(x,t,u,u_t,u_x,u_{xx})$) a two-level difference scheme with a predictor-corrector algorithm will be presented. The quadrature error of the solution method is of fourth order. The solution method satisfies necessary (von Neumann) stability conditions for local stability.

Some applications of the solution method to membranes with hysteresis type stress-strain laws will be presented. The computations exhibit the nature of the propagation of stresses along the membrane. They can be used to analyze numerically the dynamic response of membranes, to predict places of possible failures and to predict permanent deformations. Together with adequate experiments such computations may be used to test the existing theories about the behavior of membranes under large stresses.

1. PROBLEM OUTLINE

Typical objectives of investigations concerned with the dynamic response of thin metal shells to blast loads are the predictions of large and permanent deformations. This means that the corresponding equations of motion cannot be linearized assuming "small deflections" nor can the stress-strain law be assumed linear. In order to keep the computation of such deformations in spite of these conditions as simple as possible a membrane theory may be applied by assuming that the bending strength of the shell is negligible. The present paper deals with calculations within the scope of such a membrane theory. It is an open question to what extent this model can be used to describe the behavior of a real shell. This problem will not be discussed here. It could be clarified by comparison of the computed deformations with corresponding experimental values. Independently of such experiment, the calculations presented may be of interest as an example for the numerical treatment of a system of non-linear partial differential equations. The algorithm developed does not utilize the special form of the equations and can be applied to any hyperbolic second order system with two independent variables.

Examples for the application of the algorithm are presented in Section 4. They show the necessity to keep in close touch with the mechanical interpretation of the mathematical expressions used. In the case considered here the discretized equations of motion of the membrane can be interpreted as equations of a mechanical model of the membrane. This interpretation gives a better insight into the behavior of the solutions and permits to select an adequate discretization process.

About the membrane we make the following assumptions throughout the paper:

(a) the membrane is perfectly flexible
(b) the membrane is infinitely long in one direction, say z;
(c) the loads are independent of z.

We note that with the assumptions (b) and (c) we are actually reducing the problem to the motion of a perfectly flexible wire in a plane. This limitation arised naturally from applications which will not be discussed here. The elastic properties of a membrane which satisfies (a), (b) and (c) can be described by a single stress-strain function $\sigma(\varepsilon)$. In view of the applications this function was assumed as general as possible. Of particular interest was a history dependent stress-strain function which is a combination of Hooke's linear and Bell's parabolic laws. An example of such a function is shown in Figure 1. (The quantity A in $\sigma = A\sqrt{\varepsilon}$ is equal to $\alpha(1-T/T_m)$, where α is essentially constant, T is the temperature of the material and T_m is its melting temperature. For details see Ref. 1.)

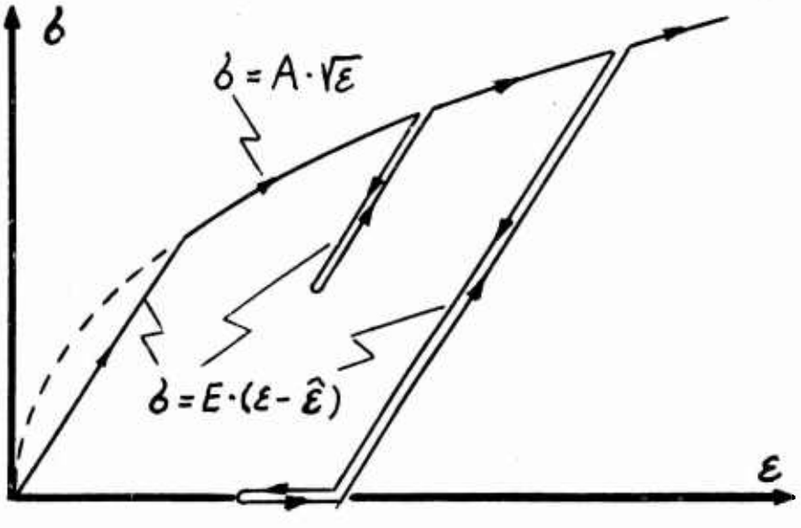


Figure 1. Stress-strain history

One consequence of the dependence of $\sigma(\epsilon)$ on the stress history is that in general for each element of the membrane we have a different $\sigma(\epsilon)$ at any given time, even if the constants A and E are the same for all elements. Hence if the problem is solved under these conditions, the inclusion of more complicated stress-strain relationships requires only trivial changes of the computing process. (More complicated laws may be assumed if the membrane is combined of parts of different materials or heated during the motion.)

Typical loads on the membrane are time dependent pressures. The computation process presented here can be used, however, for the treatment of more general loads, too. Such loads may be needed, for instance, to obtain the motion of a membrane (or wire) hit by a projectile.

2. EQUATIONS OF MOTION

We are mainly interested in the motion of a membrane which is fixed at two boundaries and flat at the beginning of the motion. We chose, therefore, a coordinate system adapted to that situation. (This choice does of course not exclude other initial configurations.) We assume that the fixed boundaries are perpendicular to the x,y-plane and pass through the x-axis at x = a and x = b. If we think of the problem in terms of an extensible wire, then the wire occupies at rest the segment of the x-axis between x = a and x = b. Loads on the wire are within the x,y-plane only. The deformation of the membrane can be described using this coordinate system by two functions, u(x,t) and v(x,t) as shown in Figure 2. Thus, a point of the membrane which is at time zero

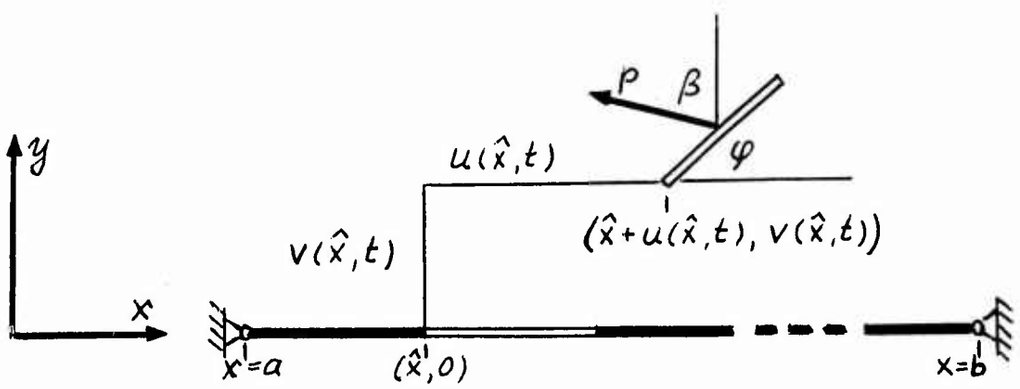


Figure 2. Position of an element at time zero and t

located at $x = \hat{x}$, $y = 0$, has at time t the coordinates $x = \hat{x} + u(\hat{x},t)$, $y = v(\hat{x},t)$.

Denoting time derivatives by dots and derivatives with respect to x by apostrophes we obtain the following equations of motion:

$$\ddot{u} = \frac{1}{\rho_o} (\sigma \cdot \cos \varphi)' - \frac{p}{\rho_o \cdot c} \cdot s' \cdot \sin \beta, \tag{1}$$

$$\ddot{v} = \frac{1}{\rho_o} (\sigma \cdot \sin \varphi)' + \frac{p}{\rho_o \cdot c} \cdot s' \cos \beta, \tag{2}$$

$$s' = \sqrt{(1+u')^2 + v'^2}, \tag{3}$$

$$\sin \varphi = v'/s', \tag{4}$$

$$\cos \varphi = (1+u')/s', \tag{5}$$

$$\varepsilon = \frac{2u' + u'^2 + v'^2}{1 + s'} + \varepsilon_o \cdot s', \tag{6}$$

where

$\rho_o(x)$ is the density of the membrane at the beginning of the motion,

c is the thickness of the membrane,

ε_o is the strain in the membrane when it rests between $x = a$ and $x = b$,

$p(x,t)$ is a pressure type load acting in direction β. (For normal pressure $\beta = \varphi$.)

A detailed derivation and discussion of these equations is given in Ref. 2.

In addition to these equations we assume that a stress-strain relationship $\sigma(\varepsilon)$ of the type discussed in Section 1 is given.

Equations (1) and (2) can be replaced by the following quasi-linear Equations (1*) and (2*), respectively:

$$\ddot{u} = u'' \frac{1}{\rho_o} \left[\frac{\sigma}{s'} \cdot \sin^2 \varphi + (1+\varepsilon_o) \frac{d\sigma}{d\varepsilon} \cos^2 \varphi \right] + \tag{1*}$$

$$+ v'' \frac{1}{\rho_o} \left[- \frac{\sigma}{s'} + (1+\varepsilon_o) \frac{d\sigma}{d\varepsilon} \right] \sin \varphi \cdot \cos \varphi - \frac{p}{\rho_o c} s' \cdot \sin \beta,$$

$$\ddot{v} = u'' \frac{1}{\rho_o} \left[- \frac{\sigma}{s'} + (1+\varepsilon_o) \frac{d\sigma}{d\varepsilon} \cdot \sin \varphi \cdot \cos \varphi \right] + \tag{2*}$$

$$+ v'' \frac{1}{\rho_o} \left[\frac{\sigma}{s'} \cos^2 \varphi + (1+\varepsilon_o) \frac{d\sigma}{d\varepsilon} \sin^2 \varphi \right] + \frac{p}{\rho_o c} s' \cos \beta.$$

In order to simplify the notation let w be a vector function. (In the present case $w = \binom{u}{v}$.) The equations of motion are then of the type

$$\ddot{w} = f(x,t,w,\dot{w},w',w''). \tag{7}$$

The slopes α of the characteristics of (7) in the x,t-plane satisfy the equation

$$\det (\alpha^2 D - I) = 0, \tag{8}$$

where

$$D = \partial f/\partial w'' \tag{9}$$

is the Jacobian of f and I is the unity matrix. If all roots of Equation (8) are real we call Equation (7) totally hyperbolic (Ref 3, p.175).

The conditions for total hyperbolicity of Equation (7) can be expressed in case of two-dimensional w conveniently in terms of the trace Θ and determinant Δ of the Jacobian D. In terms of these quantities Equation (8) becomes namely

$$(\alpha^2\Theta)^2 \cdot \frac{\Delta}{\Theta^2} - \alpha^2\Theta + 1 = 0 \tag{10}$$

and this equation has four real roots if and only if

$$\Theta \geq 0$$

and $\tag{11}$

$$\Theta^2/4 \geq \Delta \geq 0.$$

Computing Δ and Θ from Equations (1*) and (2*) we obtain

$$\Theta = \frac{\sigma}{\rho_o}\gamma(1+\gamma), \tag{12}$$

$$\Delta = \Theta^2 \cdot \frac{\gamma}{1+\gamma} \tag{13}$$

with

$$\gamma = \frac{\sigma}{(1+\epsilon_o) \cdot s' \cdot d\sigma/d\epsilon}. \tag{14}$$

These quantities Θ and Δ satisfy obviously the conditions (11). Hence the equations of motion, (1*) and (2*), are totally hyperbolic. The slopes of the characteristics are

$$\alpha^2_{min} = \frac{2}{\Theta}\left(1 + \sqrt{1 - 4(\Delta/\Theta^2)}\right)^{-1} = \frac{\rho_o}{(1+\varepsilon_o)\, d\sigma/d\varepsilon} \tag{15}$$

and

$$\alpha^2_{max} = \frac{\Theta}{2\Delta}\left(1 + \sqrt{1 - 4(\Delta/\Theta^2)}\right) = \frac{\rho_o \cdot s'}{\sigma} . \tag{16}$$

α_{min} is the inverse of the longitudinal wave velocity and α_{max} is the inverse of the transversal wave velocity (Ref. 4).

3. NUMERICAL SOLUTION

A numerical solution of the equations of motion of an extensible string has been considered by Cristescu (Ref. 4). He uses a characteristics method and presents examples for a string with a linear stress-strain law (Ref. 5). In this paper we present another numerical method which has the advantage of simplicity and which may be readily applied to any totally hyperbolic second order system with two independent variables. It does not require a transformation of the equations in characteristics form nor a transcription in form of a system of first order equations. A new feature of the numerical scheme is the employment of a two level finite difference scheme for a system of second order equations. This has the advantage that the sizes of the time steps can be changed easily during the computation process, thus reducing computing time and enhancing convergence and accuracy of the method.

We now give a short description of the numerical solution method. A detailed description and analysis of the method is given in Ref. 6.

The solution method is based on the following interpolation formulae for the vector functions w and $\dot{w}$:

$$w(x,t+k) = w(x,t) + k\dot{w}(x,t) + \frac{1}{3}k^2\,\ddot{w}(x,t) + \\ + \frac{1}{6}k^2\,\ddot{w}(x,t+k) - \frac{1}{24}k^4\,\dddot{w}(x,t+\theta k), \tag{17}$$

with $0 \le \theta \le 1$, and

$$\dot{w}(x,t+k) = \dot{w}(x,t) + \frac{1}{2}k\cdot\ddot{w}(x,t) + \\ + \frac{1}{2}k\,\ddot{w}(x,t+k) - \frac{1}{12}k^3\,\dddot{w}(x,t+\hat{\theta}k), \tag{18}$$

with $0 \le \hat{\theta} \le 1$.

For the numerical realization of these formulae we establish a rectangular computing grid in the x,t-plane with grid points (x_i,t_j) defined by

$$h = (b-a)/n,$$
$$x_i = a + i \cdot h, \quad (i=0,1,2,\ldots,n) \tag{19}$$

and

$$t_o = 0$$
$$t_j = \sum_{r=1}^{j} k_r, \quad (j=1,2,\ldots). \tag{20}$$

This grid is equidistant in the x-direction and it has variable time steps k_r. The step k_{r+1} we chose such that the estimated domain of dependence for every point (x_i,t_{r+1}), $i=1,2,\ldots,n-1$ is located for $t=t_r$ between the points (x_{i-1},t_r) and (x_{i+1},t_r). Values computed at the grid points we denote by corresponding indices, i.e., $w(x_i,t_r) = w_{i,r}$, $\dot{w}(x_i,t_r) = \dot{w}_{i,r}$, etc.

The computing step which furnishes values of w and $\dot{w}$ at the grid points with $t = t_{r+1}$ from those at $t = t_r$ is a predictor-corrector type algorithm and may be described by the following four substeps.

Substep 1

For $t = t_r$ we obtain approximations to the space derivatives

$$W'_{i,r} = \frac{1}{2h}(W_{i+1,r} - W_{i-1,r}),$$
$$W''_{i,r} = \frac{1}{h^2}(W_{i+1,r} - 2W_{i,r} + W_{i-1,r}). \tag{21}$$

Using these approximations and Equation (7) we compute the approximations $\ddot{W}_{i,r}$. We also compute with Equation (15) α_{min} at every grid point (x_i,t_r), $i=1,\ldots,n-1$ and establish the new time interval k_{r+1} by

$$k_{r+1} = h \cdot \min_{1 \le i \le n-1} \left|\alpha_{min}(x_i,t_r)\right| \tag{22}$$

Substep 2 ("Predictor").

We compute by extrapolation approximations to $W_{i,r+1}$ and $\dot{W}_{i,r+1}$:

$$\begin{aligned} {}^{*}W_{i,r+1} &= W_{i,r} + k_{r+1}\dot{W}_{i,r} + \frac{1}{2}k_{r+1}^{2}\ddot{W}_{i,r}\,, \\ {}^{*}\dot{W}_{i,r+1} &= \dot{W}_{i,r} + k_{r+1}\ddot{W}_{i,r}. \end{aligned} \tag{23}$$

Substep 3.

We compute approximations to the space derivatives for $t = t_{r+1}$, using the latest approximations of $W_{i,r+1}$:

$$\begin{aligned} {}^{*}W'_{i,r+1} &= \frac{1}{2h}\left({}^{*}W_{i+1,r+1} - {}^{*}W_{i-1,r+1}\right), \\ {}^{*}W''_{i,r+1} &= \frac{1}{h^{2}}\left({}^{*}W_{i+1,r+1} - 2\,{}^{*}W_{i,r+1} + {}^{*}W_{i-1,r+1}\right). \end{aligned} \tag{24}$$

Corresponding ${}^{*}\ddot{W}_{i,r+1}$ are then computed with Equation (7) using these values.

Substep 4 ("Corrector").

The approximations ${}^{*}W_{i,r+1}$ and ${}^{*}W_{i,r+1}$ are corrected by computing new approximations:

$$\begin{aligned} W_{i,r+1} &= {}^{*}W_{i,r+1} + \frac{1}{6}k_{r+1}^{2}\left({}^{*}\ddot{W}_{i,r+1} - \ddot{W}_{i,r}\right), \\ \dot{W}_{i,r+1} &= {}^{*}W_{i,r+1} + \frac{1}{2}k_{r}\left({}^{*}\ddot{W}_{i,r+1} - \ddot{W}_{i,r}\right). \end{aligned} \tag{25}$$

The boundary conditions enter the computations at appropriate substeps for $i = 0$ and $i = n$.

If the computations stop at Substep 4, we have a predictor-corrector type computing scheme. A possible variation of the scheme can be obtained by an iteration between Substeps 3 and 4. This amounts to an iterative solution of Equations (17) and (18) without the remainder terms.

It can be shown (Ref. 6) that von Neumann conditions for local stability are satisfied for this computing scheme if the system of Equation (7) is totally hyperbolic, f differentiable with respect to all its arguments and

$$k_{r}/h \leqq \sqrt{3} \cdot \min_{a < x < b}\left(\alpha_{\min}(x,t_{r})\right). \tag{26}$$

Hence, by chosing the time interval k_r in accordance with the domain of dependence, i.e., Equation (22), we remain well within this von Neumann bound.

In the present two-dimensional case we can express the stability and hyperbolicity conditions in terms of the determinant Δ and the trace Θ of the Jacobian D. The results are shown in Figure 3.

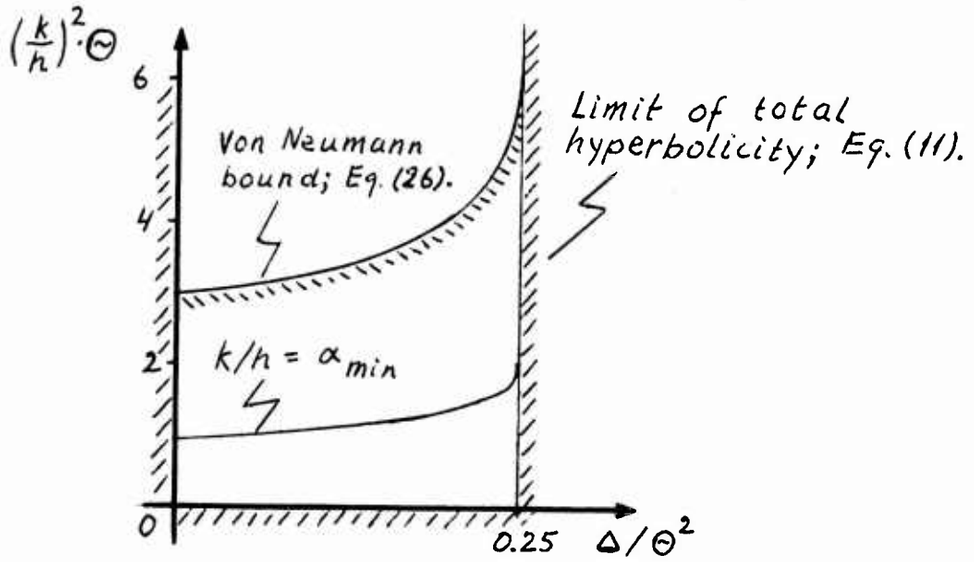


Figure 3. Local stability of computing scheme

The order of the computing scheme is 4 for W and 3 for $\dot{W}$ and it converges to the solution with refinement of the grid if the solution is smooth. The replacing of the predictor-corrector scheme by an iterative process reduces some error terms but does not increase the order of the approximation. Hence a general improvement of the solutions by iteration cannot be expected. - The computing scheme is not suitable for the treatment of motions with shock phenomena.

4. EXAMPLES

The numerical solution method outlined in the previous section is a finite difference method. It requires, therefore, a discretization of the equations of motion. The discretized equations might not be an adequate description of the continuous mechanical model for which the original equations of motion are derived. In such cases we may try to modify the discretization process. The adequacy of the final equations can be established, for instance, if we can find a mechanical model for the discrete equations. If that model is sufficiently simple it can be of great help for the understanding of the behavior of the solutions. We illustrate this situation by considering the stresses and strains for a special deformation of the membrane.

Assume that for $t = 0$ the membrane is stress free ($\epsilon_o = 0$) and exposed to a load as shown in Figure 4a. At time $t = t_1$ the load may be zero and the deformation as in Figure 4b. It is obvious that in this situation the point $x = 3$ of the membrane will have a positive acceleration (for $t = t_1$). An inspection of Equation (2*) shows that indeed for $t = t_1$

$$\ddot{v}_3 = (v_3'' \sigma_3)/(\rho_o s_3),$$

that is, $\ddot{v}_3 > 0$ if $\sigma_3 > 0$. Hence necessary for $\ddot{v}_3 > 0$ is $\epsilon_3 > 0$. (In case of a linear stress-strain law $\sigma = E\epsilon$, $\epsilon_3 > 0$ is also sufficient for positive acceleration.) Since ϵ is computed from Equation (6) with $v_3' = 0$ and $u_3' > 0$, we have simply

$$\epsilon_3 = u_3' > 0,$$

as expected.

For the discretized problem we use u- and v-values at the grid points only. Because at these points $u_i = 0$, we obtain $u_3' = 0$ and $\epsilon_3 = 0$. (The other ϵ_i are shown in Figure 4c.) Hence if the numerical scheme is applied to the deformation in Figure 4b we obtain $\ddot{v}_3 = 0$, which is obviously not correct. This demonstrates that the formally discretized equations of motion do not describe adequately the membrane model.

Searching for a more adequate numerical algorithm we may change the computation process of ϵ. Instead of using Equation (6) we compute the deformations between the grid points k and k + 1 by

$$\epsilon_{k,k+1} = (s_{k,k+1} - h)/h,$$

where $s_{k,k+1}$ is the distance between the points k and k + 1 (see Figure 4e). The strains ϵ_i at the grid points, which are needed to evaluate the right hand sides of Equations (1*) and (2*), we can then compute by averaging:

$$\epsilon_i = \frac{1}{2} (\epsilon_{i-1,i} + \epsilon_{i,i+1}).$$

The resulting ϵ_i are shown in Figure 4d. Obviously these values are better than those of Figure 4c, showing a qualitatively correct behavior for the deformation considered. However, objections can be raised against an averaging in case of a general stress-strain law. In the mechanical model this may amount to an averaging between materials in elastic and plastic status. Computationally we may average between two different branches of the function $\sigma(\epsilon)$.

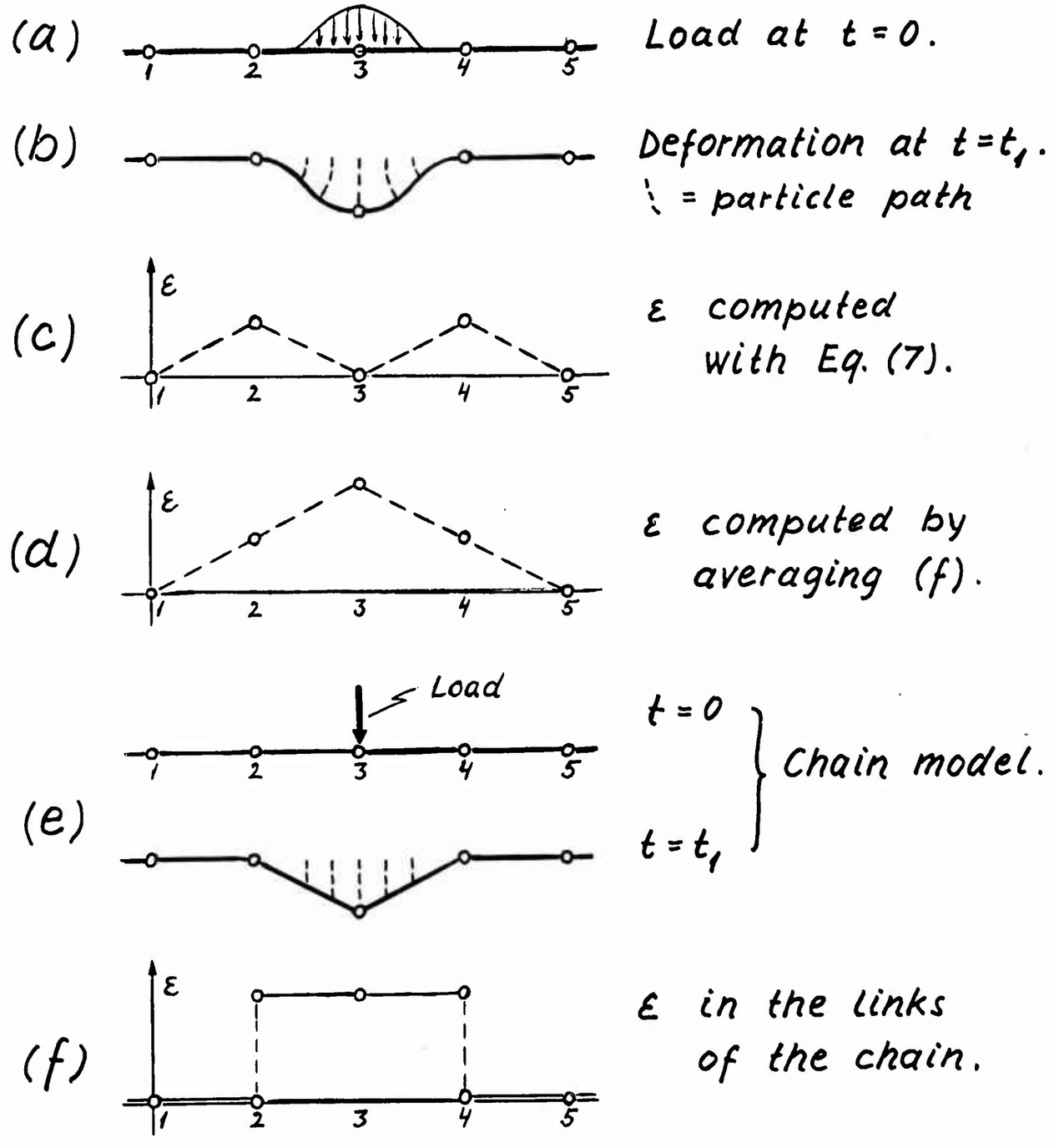


Fig. 4. Strain computations.

The last considerations suggest another possibility to handle the problem: we compute strains and stresses between the grid points as above, but do not asign specific values of σ at the grid points. (Figure 4f) This approach has a simple mechanical model, namely a chain consisting of mass free extensible and straight links and mass points attached to the hinges at the grid points. (Figure 4e) Computationally this approach amounts to a discretization of Equations (1) and (2) instead of Equations (1*) and (2*), respectively.

All three methods for the computation of ϵ and σ were tried out in the examples shown later. The first and second method proved to be numerically unstable. The instabilities appeared first at places where an unloading of the membrane took place, that is, where $\sigma(\epsilon)$ ceased to be a unique function. The computations for the chain model had no instabilities. All three models furnished practically identical results before unloading.

The first example of the computations deals with the dynamic response of a membrane to a high pressure load of short duration, as shown in Figure 5. The membrane was assumed 20cm wide, 1mm thick and having the

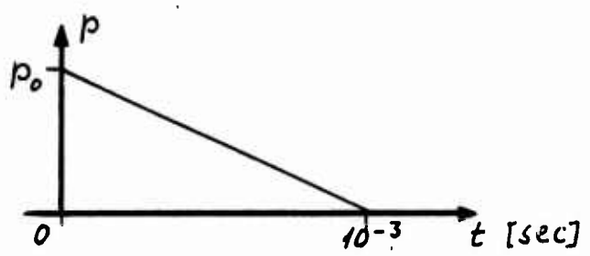


Figure 5. Pressure load

following material constants:

Density $\rho_o = 2.7\ g/cm^3$;

Linear elasticity modulus $E = 7200\ kg/mm^2$;

Parabolic elasticity modulus $A = 40\ kg/mm^2$.

(These values may correspond to an aluminum aloy.) The initial strain ϵ_o was assumed zero. The initial pressure ρ_o was assumed to be 100 technical atmospheres (= $100\ kg/cm^2$).

The deformations at various time instants are shown in Figures 6, 7 and 8 together with the corresponding stress and strain profiles. It is interesting to note that the latter profiles are almost constant for any fixed time. The reason for this is, that the stresses, induced at the boundaries of the membrane are carried by fast longitudinal waves and increased by relatively small steps as these waves travel back and forth across the membrane. The transversal waves do not carry any larger amount of stresses in this case. A practical consequence of this behavior of the membrane is that a failure can occur at any place and a

rupture is not bound to take place at the boundaries or at the point of symmetry.

As mentioned in Section 3 the numerical scheme uses variable time steps, depending on the slope of characteristics. In the example presented this slope changes as the membrane expands because of the non-linear stress-strain law. The time intervals are changed correspondingly and their sizes displayed in Figure 9, where the size of every tenth time step is plotted over the corresponding time value. Note, that at the beginning of the calculations the time intervals remain constant, which indicates that some parts of the membrane are still within the linear part of $\sigma(\epsilon)$ (see Figure 1). After all elements of the membrane have reached the parabolic region of $\sigma(\epsilon)$, the time intervals are increased in accordance with Equations (15) and (22). This increase up to a whole order of magnitude results in an essential saving of computing time. At about $t = 6.3 \cdot 10^{-3}$ sec. unloading at some parts of the membrane starts. The corresponding stress-strain law becomes linear again (Figure 1) and consequently the time interval drops to its original value.

Another example of the computations is shown in Figures 10, 11 and 12. The same membrane as in the previous example is hit by a projectile at an angle of 45°. The projectile's cross section is assumed $25 mm^2$, its density $7.8\ g/cm^3$ and its velocity 500 m/sec. The load on the membrane is in this case implemented by computing an initial velocity of those mass points which are hit by the projectile, assuming thereby a plastic collision. The mass of the points involved in the collision is increased for subsequent calculations by adding the mass of the projectile. (In the case displayed in the Figures the number of grid points was 200, i.e., they were originally 1mm apart from each other. Consequently it was assumed that two adjacent points were hit simultaneously by the projectile and to each of these mass points one half of the projectiles mass was added.) Figures 10, 11 and 12 show the positions of the membrane and the corresponding stresses and strains at various instants. As in the previous example considerable strains are found in the membrane already before the arrival of the transversal wave. Due to the nature of the load maximal stresses are in this case obtained adjacent to the impact zone and at the boundaries of the membrane. The critical instants for a possible break of the membrane are for the impact zone immediately after the impact and for the boundaries the arrival times of the transversal waves. (For the left hand boundary of about $3.5 \cdot 10^{-4}$ sec. and for the right hand boundary at about 7.10^{-4} sec.) However, as the permanent deformations in Figure 12, last picture, show, the maxima are not very pronounced and failure might be possible anywhere within the left half of the membrane.

5. SUMMARY

A novel numerical treatment of the dynamic response of a membrane has been presented. The method employs a chain model and furnishes all data necessary for a numerical analysis of the physical phenomena. It can be applied to membranes with arbitrary stress-strain laws. Experimental verification of the validity of the model for practical problems is needed.

REFERENCES

(1) James F. Bell, The Physics of Large Deformations of Crystalline Solids, Springer Tracts of Nat. Philos., Vol. 14, Springer-Verlag, New York, 1968.

(2) Aivars Celmiņš, Computation of the Motions of Extensible Strings, BRL Report, to be published.

(3) R. Courant and D. Hilbert, Methods of Mathematical Physics, Vol II, Inerscience Publishers, New York, 1962.

(4) N. Cristescu, Spatial Motion of Elastic-Plastic Strings, Journal of the Mech. and Phys. of Solids, Vol. 9, 1961, pp. 165-178.

(5) N. Cristescu, Rapid Motions of Extensible Strings, Journal of the Mech. and Phys. of Solids, Vol. 12, 1964, pp. 269-278.

(6) Aivars Celmiņš, A Numerical Method for the Solution of Totally Hyperbolic Systems of 2nd Order Partial Differential Equations, BRL Report, to be published.

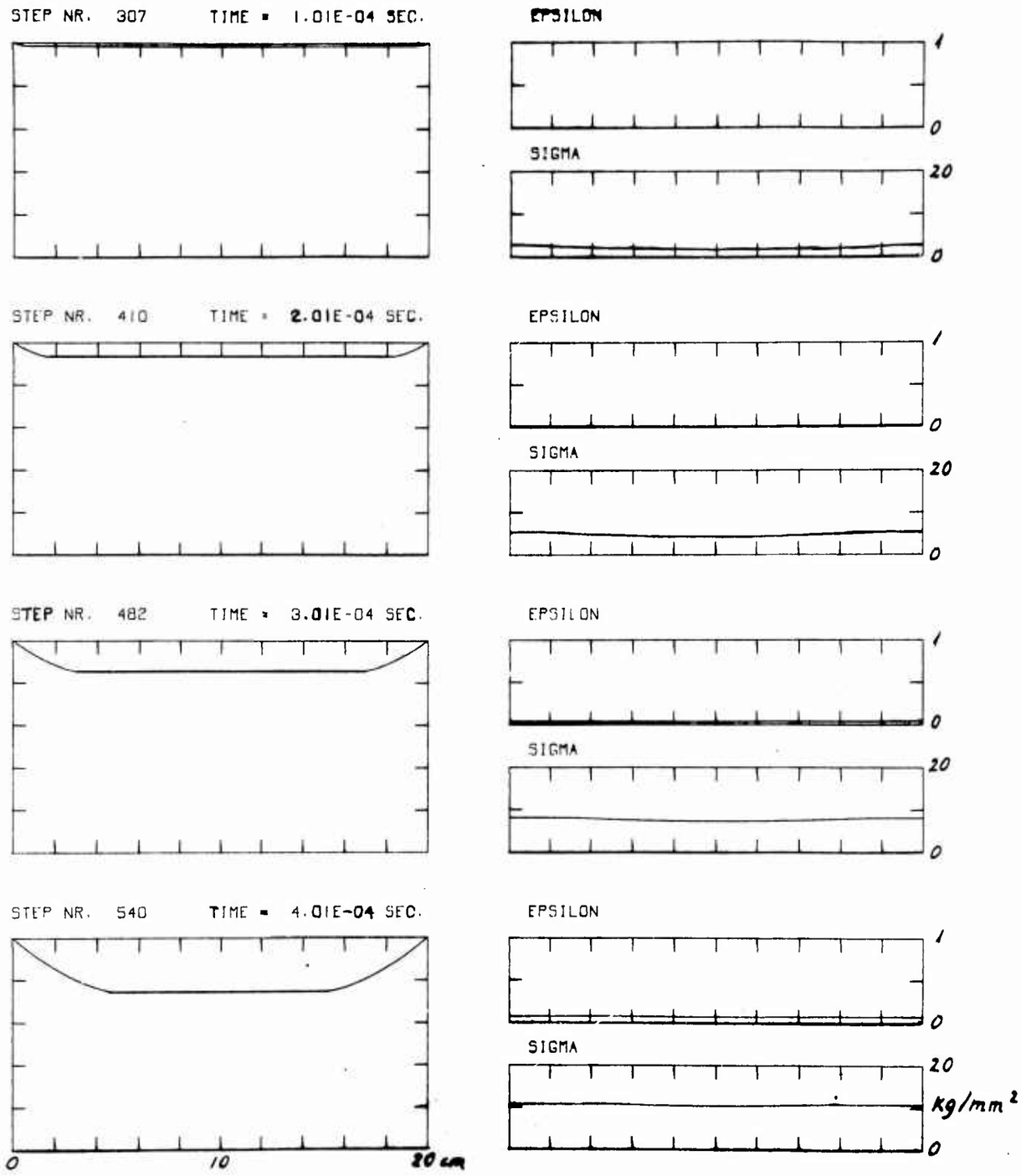


Fig. 6. Blast load.

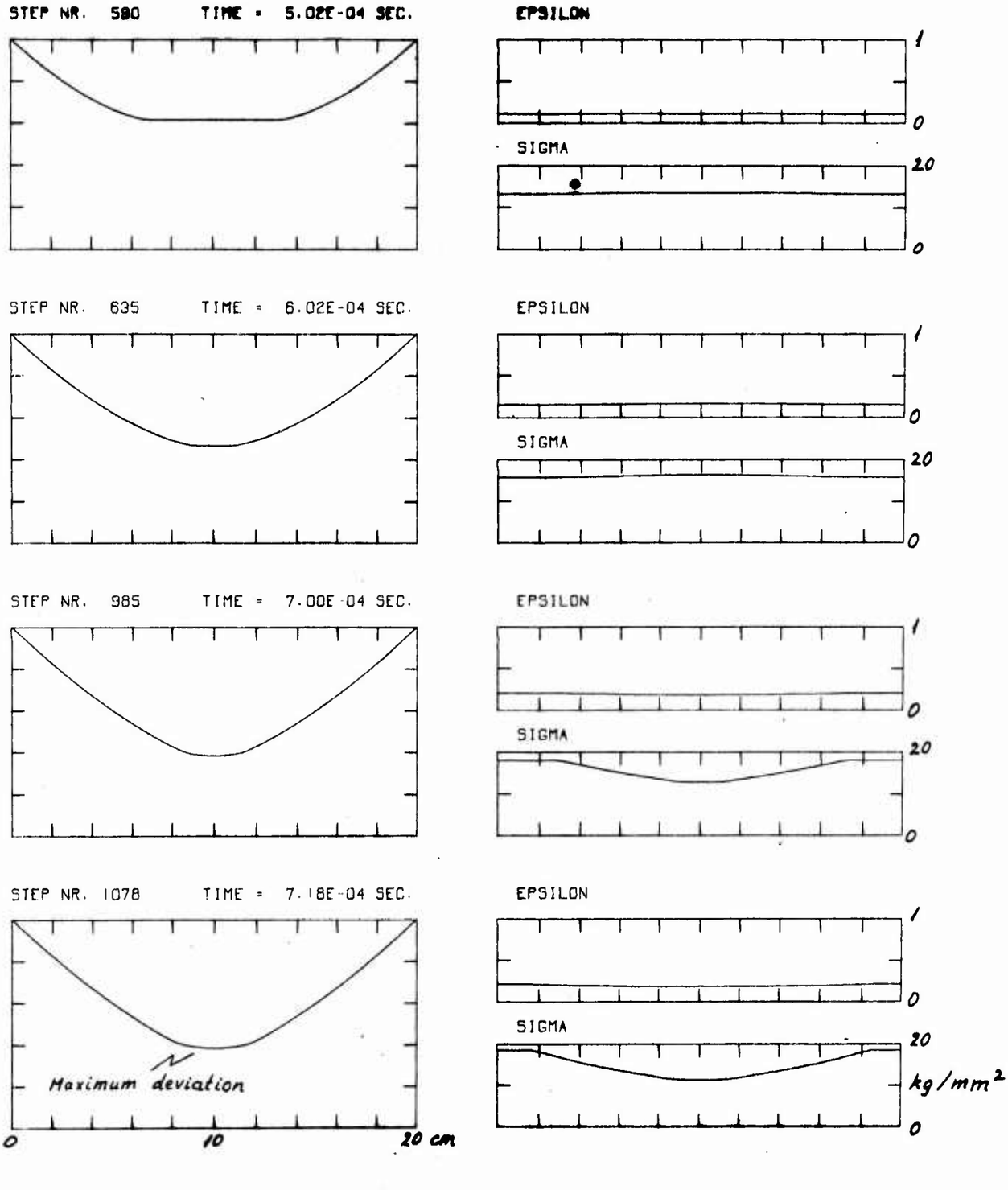


Fig. 7. Blast load.

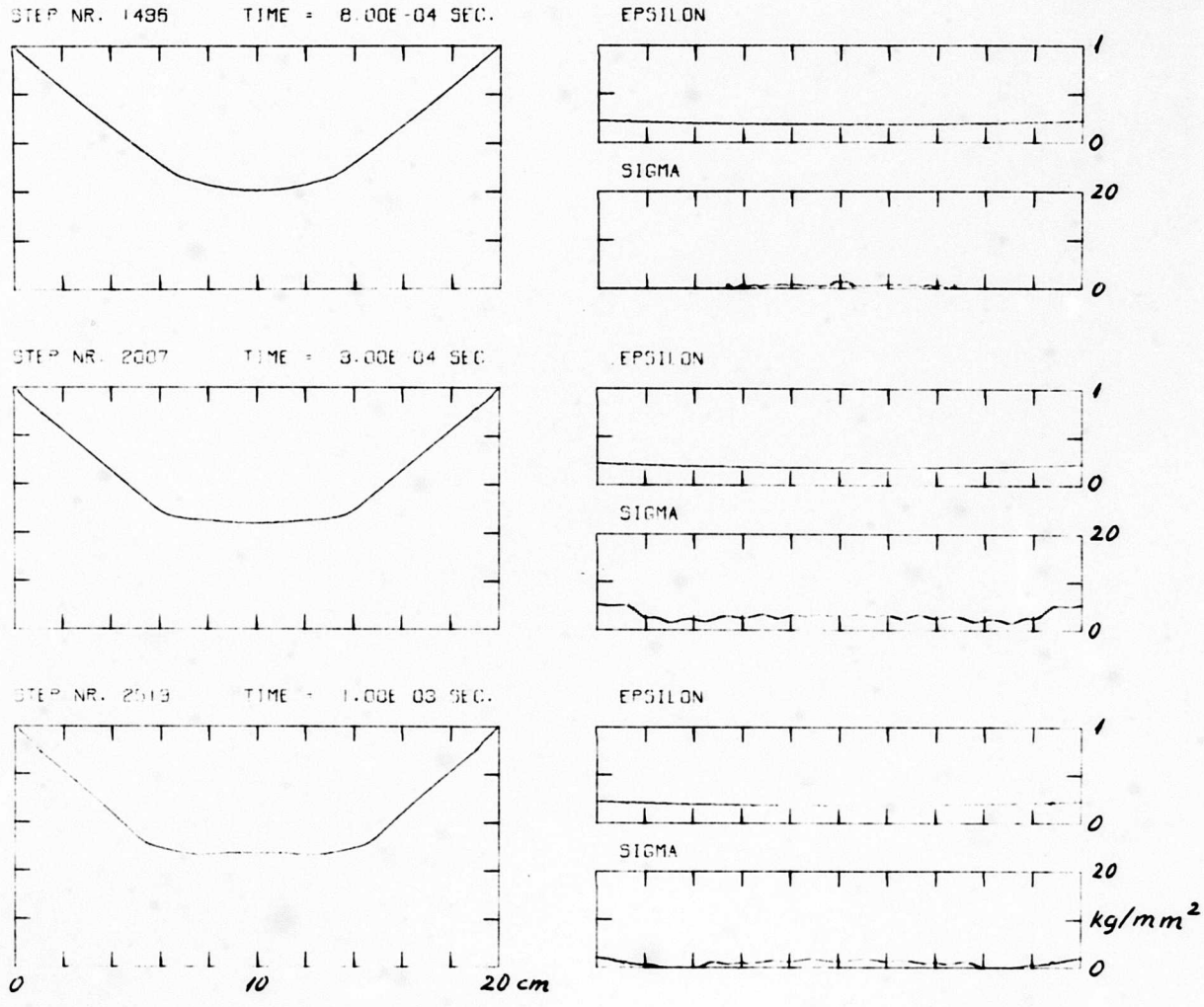


Fig. 8. Blast load.

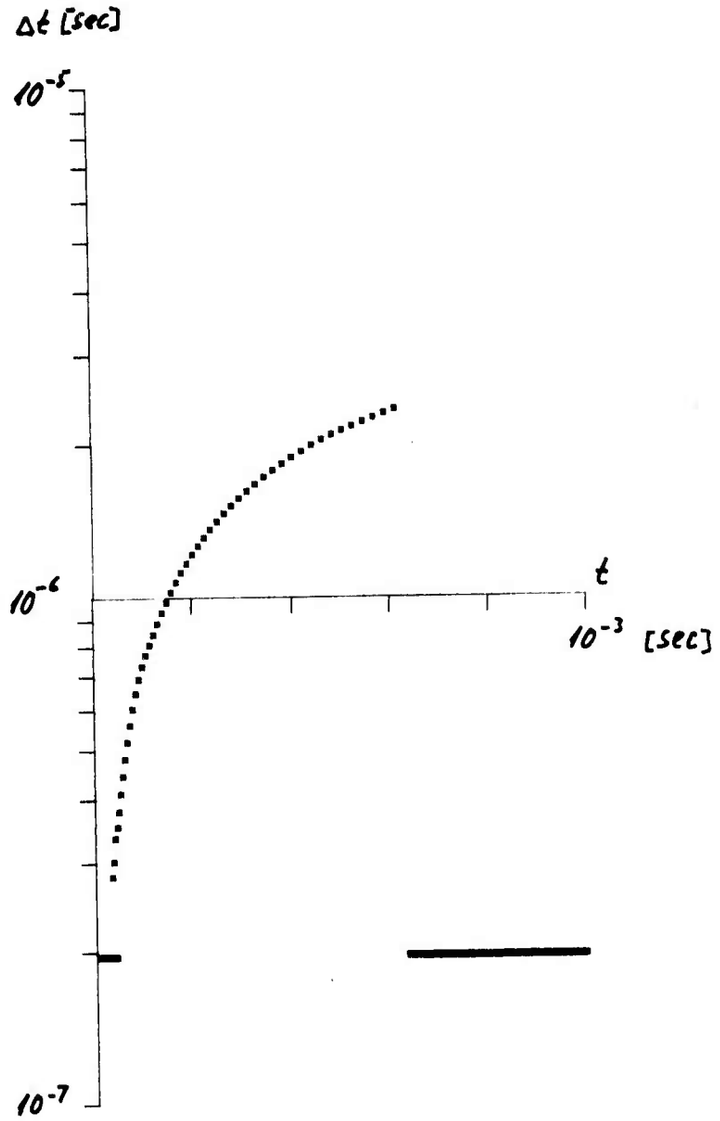


Fig. 9. Time intervals.

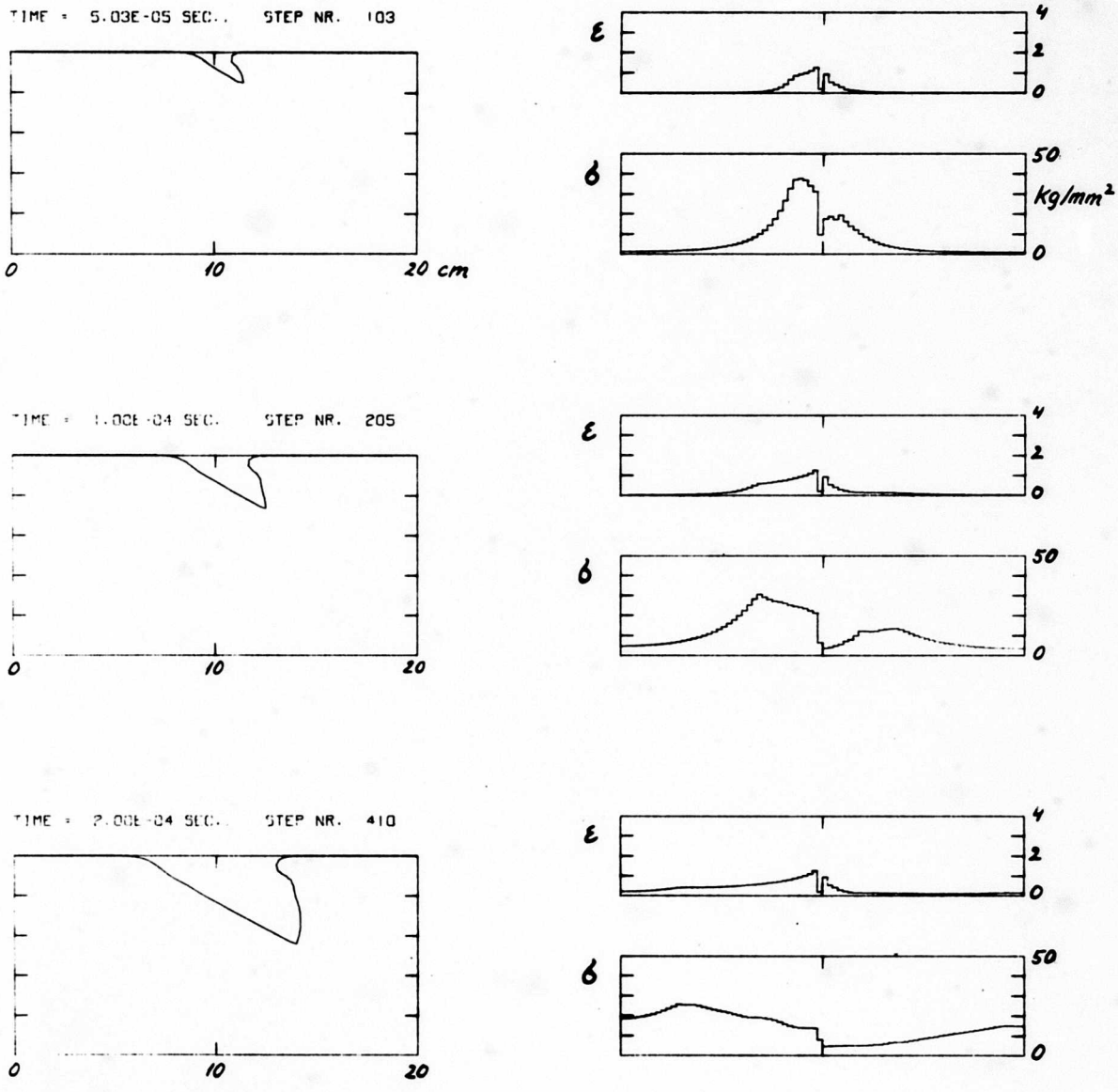


Fig. 10. Oblique impact.

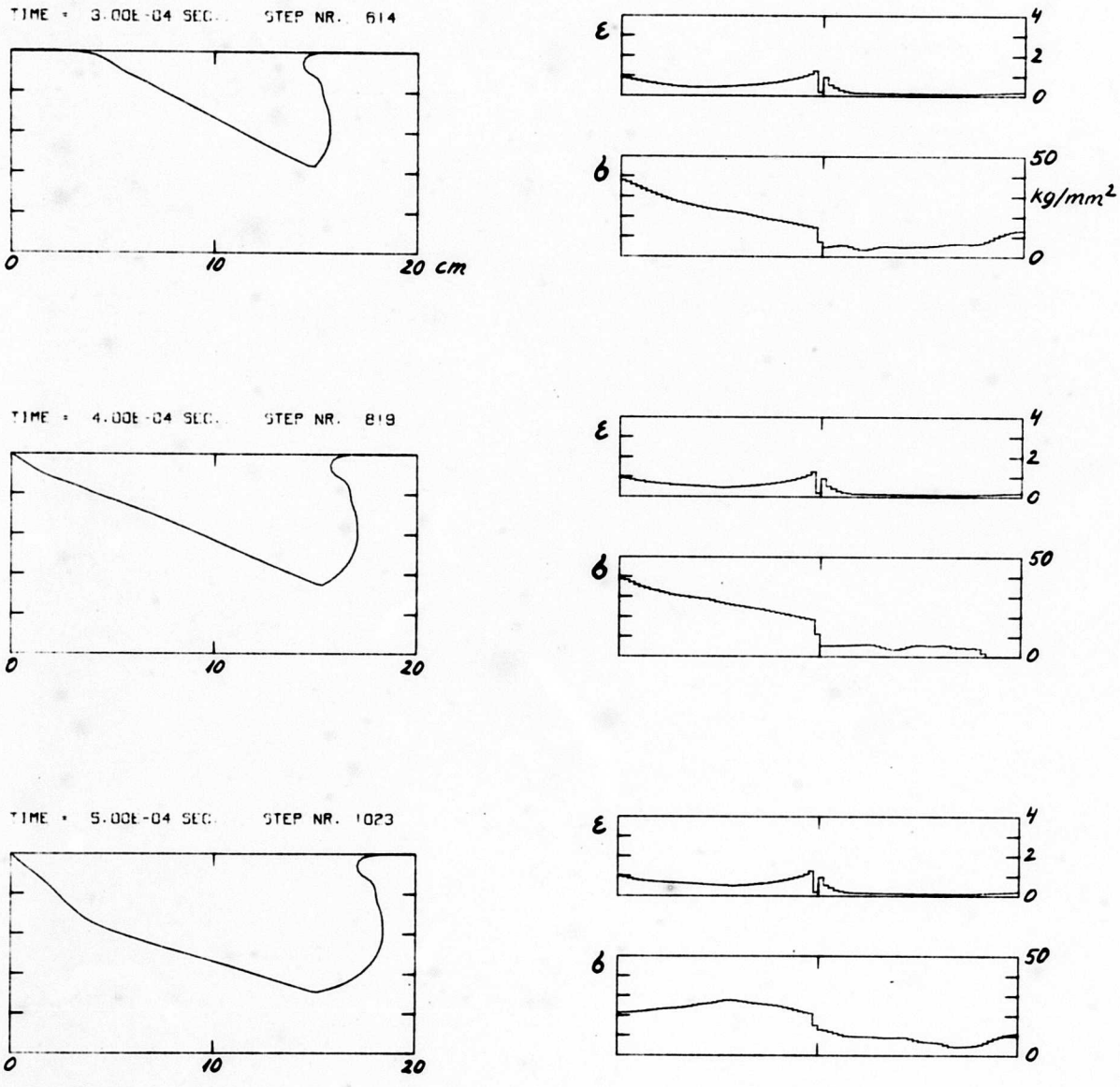


Fig. 11. Oblique impact.

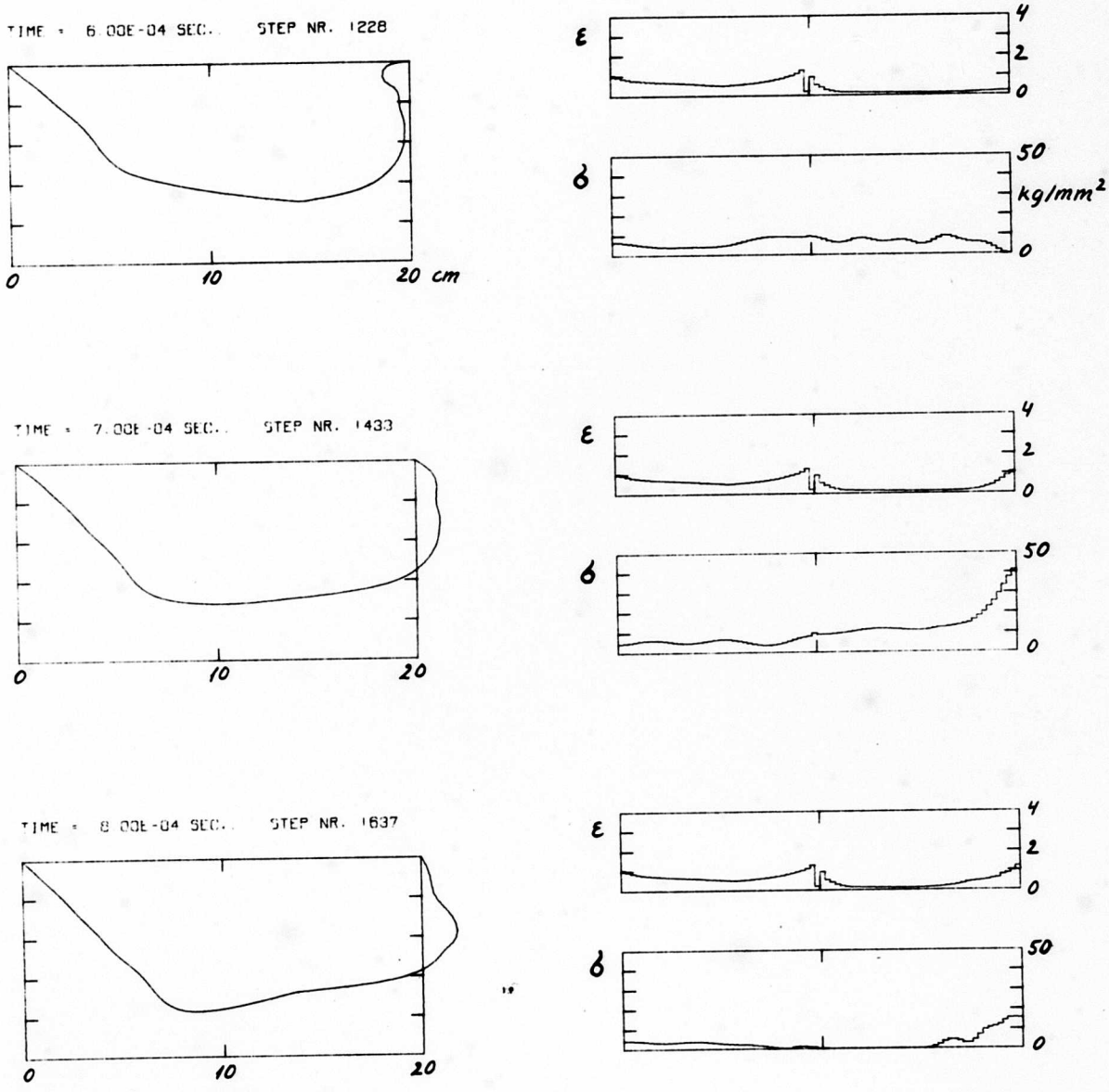


Fig. 12. Oblique impact.

BLANK PAGE

INTEGRATION OF THE TIME-DEPENDENT NAVIER-STOKES EQUATIONS IN TWO DIMENSIONS*

S.C.R. Dennis

Mathematics Research Center, U.S. Army

Madison, Wisconsin **

Gau-Zu Chang

University of Western Ontario

ABSTRACT

In this paper a method of numerical integration of the time-dependent Navier-Stokes equations for two-dimensional flows is considered. The method is restricted to classes of motion defined in a rectangular domain. However, by use of a suitable transformation this includes a large class of problems describing the flow past cylinders in an otherwise unbounded field. The formulation of the problem in terms of the stream function and vorticity is adopted. The main object of the paper is to show that the boundary-value problem which must be solved to determine the stream function can be replaced by a spatial step-by-step integration. Some numerical illustrations of the methods used are given. These include some results for the flow past a circular cylinder.

---

* Sponsored in part by the Mathematics Research Center, U.S. Army, Madison, Wisconsin under Contract No.: DA-31-124-ARO-D-462 and in part by the National Research Council of Canada. The original talk was given by one author but material due to both is included in the report.

** On leave of absence from the University of Western Ontario.

INTRODUCTION

In recent times there has been much interest in the solution of the time-dependent Navier-Stokes equations. One reason is that, starting from some initial state, if a stable integration procedure is chosen and the integrations carried on for sufficiently long time, a steady state solution may be approached. Also, time-dependent integrations may be used to predict flow configurations which are basically unsteady for all values of the time, no matter how large. The mathematics of both types of problems is essentially the same and may be described as follows.

Assuming incompressibility of the fluid, the Navier-Stokes equations can be written in the usual dimensionless form

$$\partial \underset{\sim}{v}/\partial t + (\underset{\sim}{v}.\nabla)\underset{\sim}{v} = -\text{grad } p + R^{-1}\nabla^2 \underset{\sim}{v}, \tag{1}$$

where $\underset{\sim}{v}$ and p are respectively the dimensionless velocity vector and the pressure, t is the time, and R the Reynolds number. For two-dimensional configurations in the (x,y)-plane, the equation of continuity div $\underset{\sim}{v} = 0$ can be satisfied by introducing the dimensionless stream function $\psi(x,y,t)$, related to the velocity components (u,v) by the equations

$$u(x,y,t) = \partial\psi/\partial y, \quad v(x,y,t) = -\partial\psi/\partial x \ . \tag{2}$$

Also, the equation which results from eliminating the pressure from (1) can be expressed in terms of the dimensionless scalar vorticity $\zeta(x,y,t)$, viz. the nonzero component of curl $\underset{\sim}{v} = (0,0,\zeta)$. Thus it is found that the equations governing these two quantities are

$$\nabla^2\psi + \zeta = 0 \tag{3}$$

$$\frac{\partial \zeta}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \zeta}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \zeta}{\partial y} = R^{-1}\nabla^2\zeta, \tag{4}$$

where $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$. The necessary boundary conditions are as

follows. Where the fluid is in contact with a solid boundary designated by the curve C,

$$\psi \text{ and } \partial\psi/\partial n \text{ are known on C,} \tag{5}$$

where $\partial/\partial n$ is differentiation normal to C. If the flow is enclosed, as in the case of convection in a closed cell, these are the only conditions. If it is open, as in the case of flow past a cylinder in an unbounded fluid, conditions at infinity must be imposed. Generally these reduce to the fact that the asymptotic behavior of both ψ and ζ is known as infinity is approached. However, the important point in both problems is that two conditions are prescribed for ψ on C and none for ζ.

This latter feature tends to make the integration procedure more difficult, or slower, than it might otherwise be. The source of slowness is that the solution of the Poisson-type equation (3) is a boundary-value problem for a given step in time and has to be solved by time-consuming boundary-value techniques, usually iterative methods. The solution of equation (4) on the other hand, is a marching (or step-by-step) problem in time. An integration of the pair of equations proceeds as follows. To fix ideas consider the open field type of problem in which the domain of the solution is a region D between a closed contour C on which the conditions (5) hold and an outer contour C_1, on which boundary conditions for both ζ and ψ may be supposed known. Suppose at time t both ζ and ψ are known at all points of a finite-difference grid covering D and also at all necessary points of C and C_1. Then the integration through time $t + \Delta t$ consists of the successive operations:

(i) By forward integration of equation (4), $\zeta(x,y,t + \Delta t)$ is determined from $\zeta(x,y,t)$ at all grid points in D and on C_1, but not on C since no

condition for ζ is known there.

(ii) From $\zeta(x,y,t + \Delta t)$ at grid points within D, $\psi(x,y,t + \Delta t)$ is determined from equation (3) as a boundary-value problem with ψ known on C and C_1.

(iii) The (as yet unused) condition for $\partial\psi/\partial n$ on C is now used to calculate $\zeta(x,y,t + \Delta t)$ on C from computed values of $\psi(x,y,t + \Delta t)$ at grid points in the neighborhood of C. The solution at time $t + \Delta t$ is thus completed everywhere. Successive applications of the whole process determines the solution at any subsequent time.

The main object of the present paper is to attempt to dispense with the boundary-value problem in step (ii) of the above procedure in certain classes of problem in which the domain of integration is a rectangular region. The open field type of problem mentioned above falls into such a class, for by a suitable conformal transformation the domain of the two-dimensional flow past cylinders of various shapes can be mapped on to a semi-infinite rectangle. It will be shown that for this problem we can replace the boundary-value problem in step (ii) of the above procedure by a step-by-step integration which permits the stream function to be determined by direct methods. In fact, the solution of equation (3) is re-formulated as a step-by-step problem in a single space variable.

The basis of the method for the open field problem is the following. Suppose step (i) of the above procedure has been carried out. It is shown that certain necessary conditions involving integrals of $\zeta(x,y,t + \Delta t)$ evaluated throughout the region D must at this stage be satisfied. The satisfaction of these conditions allows $\zeta(x,y,t + \Delta t)$ to be determined on the boundary C entirely from the internal solution for $\zeta(x,y,t + \Delta t)$

and without any determination of the internal solution for $\psi(x,y,t + \Delta t)$. The solution for $\zeta(x,y,t + \Delta t)$ is then complete. Since this function is now known on C as well as in D, it is now possible to integrate equation (3) subject to the conditions (5) by a step-by-step procedure in space, since clearly sufficient boundary conditions are given to formulate a step-by-step approach. This procedure determines $\psi(x,y,t + \Delta t)$ on C_1 as part of the solution and this must come out in accordance with the known boundary condition for ψ on C_1, giving an adequate check on the procedure. In the latter sense the method is useful in the open field problem because, in fact, the precise boundary condition for ψ on C_1 at a given time is not an easy one to specify. This difficulty, together with the objective of reducing computer time in time-dependent studies of the Navier-Stokes equations, has been the main motivation for the investigation. However, it is not yet known how far the latter objective has been achieved.

Although major attention is given to the open field problem of flow past a cylinder, the method is not restricted to this problem. As an illustration another type of problem, that of convection in a closed rectangular cell, is considered. Admittedly the method is less satisfactory here although the same objective, construction of a step-by-step integration of equation (3), is achieved. The two classes of problem are, however, very different in character and illustrate different features of the general method. In both classes the procedure is based on the reduction of equation (3) to a set of ordinary differential equations in one space variable. This is done by standard Fourier analysis. The resulting ordinary differential equations are then solved by step-by-step methods. In the open field problem the integration range is, in theory, infinite. In the closed cell problem it is finite. In both cases a special method

has to be constructed for solving these differential equations, owing to the nature of their solutions. Some difficulty might indeed be expected if one attempts to solve a Poisson equation, essentially of elliptic type, by step-by-step methods. The special technique used for the solution is explained and illustrated by a numerical example over a finite range of integration.

As an illustration of the open field problem, some results for the time-dependent integration of the Navier-Stokes equations for flow past a circular cylinder at Reynolds numbers 7, 10, 20, and 40 are given. Some of the results have been computed on the CDC 3600 at the University of Wisconsin and some at the University of Western Ontario. Actually, only details of the final steady-state solutions are given here, the main object being to compare them with known results in the literature. The comparison is good on the whole, indicating that the method gives satisfactory results.

PROBLEMS CONSIDERED AND REPRESENTATION BY DIFFERENCE EQUATIONS

A typical problem of flow in a closed rectangular cavity is indicated in Figure 1. At time $t < 0$ the walls OX, XY, YZ and ZO of the cavity are at rest and there is no flow. At time $t \geqslant 0$ the upper wall YZ is moved with constant velocity $u = -1$. All quantities are assumed to be dimensionless. The Reynolds number can be based on the actual length of one of the walls, say d, and the magnitude of the actual velocity of the upper wall U. Thus $R = Ud/\nu$, where ν is the coefficient of kinematical viscosity. The boundary C mentioned in the introduction is the boundary of the rectangle and the domain D its interior.

Boundary conditions for the stream function are obtained from the equations (2). Thus for $t < 0$, $\psi(x,y,t) = 0$ within D and on its boundary

C. For $t \geq 0$ we have

$$\left.\begin{aligned} &\text{On OX: } \psi = \partial\psi/\partial y = 0 \\ &\text{On XY: } \psi = \partial\psi/\partial x = 0 \\ &\text{On YZ: } \psi = 0,\ \partial\psi/\partial y = -1 \\ &\text{On ZO: } \psi = \partial\psi/\partial x = 0\ . \end{aligned}\right\} \qquad (6)$$

If the solution is started from the initial state of rest it may be continued indefinitely subject to the conditions (6). For large enough time the solution will generally tend to a steady state in which all quantities are independent of time. Equations (6) give the boundary conditions for the steady problem. Solutions to the time-dependent and steady problems for various rectangles, using numerical methods, have been given in the literature. The time-dependent problem has been considered by Greenspan, Jain, Manohar, Noble and Sakurai[1]. Steady-state solutions have been given by Kawaguti[2], Simuni[3], Mills[4] and Burggraf [5]. Both types of problems have recently been considered by Zabransky 6 .

Next consider the open field problem. A typical problem is that of a cylinder of infinite length initially at rest in an infinite, motionless, fluid. At time $t = 0$ it starts to move, and subsequently continues to move, with constant velocity in a direction perpendicular to its axis. The fluid at large enough distances is assumed to remain undisturbed. Actually we shall adopt the following equivalent formulation. At $t < 0$, the cylinder and fluid are moving together with velocity $u = 1$ parallel to the fixed x-axis. At $t = 0$ the cylinder is brought to rest and subsequently remains at rest. The fluid at large enough distances from the cylinder is assumed to remain undisturbed with velocity $u = 1$ for all t. If C denotes the contour of the cylinder and D the infinite domain outside it then, with regard to fixed rectangular axes with origin inside C, the

boundary conditions for the stream functions are

$$\left.\begin{aligned} &t < 0: \ \psi = y \text{ throughout } D \\ &t \geqslant 0: \ \psi = \partial\psi/\partial n = 0 \text{ on } C \\ &\qquad \partial\psi/\partial y \to 1, \ \partial\psi/\partial x \to 0, \text{ as } x^2 + y^2 \to \infty. \end{aligned}\right\} \quad (7)$$

We shall restrict ourselves to flows which are symmetrical about the x-axis. Thus in practice D can be limited to the region of the upper half-plane outside the upper half of C, viz. the region above GHIJK in Figure 2. Here HIJ is the cylinder and G,K are points at infinity on the x-axis. Consider also the curvilinear coordinate system (α,β) shown in Figure 2. This is supposed to be derived from the Cartesian system by a conformal transformation of the type

$$\alpha + i\beta = F(x + iy). \quad (8)$$

The transformation is chosen so that the cylinder HIJ corresponds to $\alpha = \alpha^*$ and the parts JK, GH of the x-axis correspond to $\beta = 0$, $\beta = \pi$ respectively. The region D above GHIJK therefore corresponds to the interior of the semi-infinite rectangle shown in Figure 3.

It is easily shown that under the conformal transformation (8) equation (3) becomes

$$\nabla_1^2\psi + \zeta/H^2 = 0 \quad (9)$$

and equation (4) becomes

$$\frac{\partial\zeta}{\partial t} = H^2\left(R^{-1}\nabla_1^2\zeta + \frac{\partial\psi}{\partial\alpha}\frac{\partial\zeta}{\partial\beta} - \frac{\partial\psi}{\partial\beta}\frac{\partial\zeta}{\partial\alpha}\right). \quad (10)$$

Here

$$\nabla_1^2 = \partial^2/\partial\alpha^2 + \partial^2/\partial\beta^2$$

and

$$H^2 = (\partial\alpha/\partial x)^2 + (\partial\alpha/\partial y)^2 = (\partial\beta/\partial x)^2 + (\partial\beta/\partial y)^2.$$

The use of transformations of this kind is well established in the

literature. Thus, for example, Payne [7] and Kawaguti and Jain [8] have used the particular case

$$F(x + iy) = \log(x + iy) \tag{11}$$

for which

$$H^2 = e^{-2\alpha} \tag{12}$$

in calculations of the time-dependent flow past a circular cylinder. Here $\alpha = 0$ corresponds to the cylinder $x^2 + y^2 = 1$. Apelt [9] and Keller and Takami [10] have used the same transformation in calculations of the corresponding steady flow ($\partial\zeta/\partial t = 0$ in equation (10)). The case of steady flow past a finite flat plate occupying the region $|x| \leq 1$ of the x-axis has been considered by Janssen [11] and by Dennis and Dunwoody [12] using the particular case

$$F(x + iy) = \cosh^{-1}(x + iy)$$

for which

$$H^2 = 2/(\cosh 2\alpha - \cos 2\beta).$$

In this case $\alpha^* = 0$ corresponds to the plate and it may be noted that taking α^* equal to any positive constant gives an elliptic cylinder whose ratio of minor to major axes is $\tanh \alpha^*$. Other transformations of the same type can be employed to correspond to cylinders of various other shapes, e.g. the generalized Joukowski aerofoil.

Finally, it remains to state boundary conditions appropriate to the transformed (α,β)-plane of Figure 3. Obviously the first condition of (7) will depend upon the particular transformation used. For the other conditions of (7) we use the relation between the derivatives

$$\left.\begin{aligned} \frac{\partial\psi}{\partial\alpha} &= \frac{\partial x}{\partial\alpha}\frac{\partial\psi}{\partial x} + \frac{\partial y}{\partial\alpha}\frac{\partial\psi}{\partial y} \\ \frac{\partial\psi}{\partial\beta} &= \frac{\partial x}{\partial\beta}\frac{\partial\psi}{\partial x} + \frac{\partial y}{\partial\beta}\frac{\partial\psi}{\partial y} \end{aligned}\right\} \tag{13}$$

Provided that the derivatives of x and y with regard to α and β remain finite at $\alpha = \alpha^*$, we can therefore take

$$t \geqslant 0: \psi = \partial\psi/\partial\alpha = 0 \text{ when } \alpha = \alpha^*. \tag{14}$$

For the conditions at infinity it may be observed that for both of the particular cases cited above we can deduce, using (13), that

$$\left.\begin{aligned} t \geqslant 0: e^{-\alpha}\, \partial\psi/\partial\alpha &\to k\sin\beta \\ e^{-\alpha}\, \partial\psi/\partial\beta &\to k\cos\beta, \text{ as } \alpha \to \infty. \end{aligned}\right\} \tag{15}$$

Here k is a numerical constant such that $k = 1$ for the circular cylinder and $k = \frac{1}{2}$ for the elliptic cylinder or flat plate. For other types of cylinders, e.g. the Joukowski aerofoil already mentioned, the same conditions (15) hold and we shall take (14) and (15) to be general conditions governing the problem. As a consequence of the uniform stream condition at infinity, it follows that for all time

$$\zeta \to 0 \text{ as } \alpha \to \infty. \tag{16}$$

Lastly, since symmetrical conditions have been assumed, both ψ and ζ must vanish on GH, JK (Figure 2) for all time, viz.

$$\psi = \zeta = 0, \text{ when } \beta = 0, \pi\ . \tag{17}$$

Next, consider the formulation of these problems in terms of finite-differences. For the problem of flow past a cylinder we must limit the semi-infinite domain in Figure 3 by an artificial boundary. This is designated by the line XY and it corresponds to the curve C_1 mentioned in the introduction. For both problems we now have a finite rectangle OXYZ as the domain D of the numerical solution. We suppose this to be covered by a rectangular grid with lines parallel to the sides of the rectangle. A typical set of points on this grid is shown in Figure 4. The spacing in the two directions is not assumed necessarily to be the same and we write $r^* = h^2/h_1^2$. Only the equations (9) and (10) need be considered. The

case H = 1 will then correspond to equations (3) and (4).

Using customary central-difference approximations to second derivatives (see e.g. Fox [13]), the usual approximation to equation (9) is

$$\psi_1 + \psi_3 + r^*(\psi_2 + \psi_4) - (2 + 2r^*)\,\psi_o + h^2\zeta_o/H_o^{\,2} = 0. \qquad (18)$$

The approximation to equation (10) will depend upon whether an explicit or implicit scheme in time is adopted. This is not really relevant to the present paper and we illustrate with the simple first-order explicit scheme. Denoting the right side of (10) by L we have, at the typical point 0,

$$\zeta_o(t + \Delta t) = \zeta_o(t) + \Delta t\, L_o. \qquad (19)$$

The quantity L_o must be expressed in differences. For simplicity write

$$\partial\psi/\partial\alpha = \lambda, \quad -\partial\psi/\partial\beta = \mu. \qquad (20)$$

Then if, for example, the ζ-derivatives are expressed in central differences, we have as an approximation

$$\begin{aligned} H_o^{-2}h^2L_o = {} & (R^{-1} + h\mu_o/2)\zeta_1 + (R^{-1} - h\mu_o/2)\zeta_3 \\ & + r^*\{(R^{-1} + h_1\lambda_o/2)\zeta_2 + (R^{-1} - h_1\lambda_o/2)\zeta_4\} - R^{-1}(2 + 2r^*)\zeta_o. \end{aligned} \qquad (21)$$

These equations are sufficient to carry out the process described in the introduction. Stage (i) can be carried out at all internal grid points using (19). At boundaries where ζ is known, e.g. OX, YZ and XY (using (16) as an approximation) in Figure 3, there exists no problem. It clearly cannot be carried out at the boundary OZ in Figure 3, or at any boundary in Figure 1. Stage (ii) is carried out by solving (usually iteratively) the matrix problem defined by the equations (18) with known conditions for ψ on the boundaries. In the problem of closed cell convection we can take $\psi = 0$ on all boundaries. In the cylinder problem $\psi = 0$ on all except XY. The condition on XY requires further consideration. It is certainly incorrect to take $\psi = e^{\alpha}\sin\beta$ for the finite value $\alpha = \alpha_m$ corresponding to

XY. The simplest possibility seems to be to use the first of the conditions (15) as a slope boundary condition on XY, but the only really satisfactory way is to determine the asymptotic nature of the solution for large α. For the time-dependent problem this is difficult.

Finally, in stage (iii), ζ is calculated at any boundary where it is not known. For example, at the boundary OZ in Figure 3, the central difference approximation to the condition $\partial\psi/\partial\alpha = 0$ gives, approximately,

$$\psi_1 = \psi_3.$$

Also $$\psi_o = \psi_2 = \psi_4 = 0$$

and hence (18) gives

$$\zeta_o = -2H_o{}^2\psi_1/h^2, \tag{22}$$

which determines ζ at any point on this boundary in terms of the computed internal values of ψ. In this problem this is the only boundary at which ζ must be so determined. It is hardly necessary to give the corresponding equations for finding ζ on the other boundaries of Figure 1. Thus from the given initial state, and with time steps Δt chosen according to some suitable stability criterion, the solution of either problem can be continued indefinitely.

METHOD OF SOLUTION

The object of this paper is to consider some modifications to the method of solution of equation (9). The cylinder problem is particularly suitable. Since $\psi = 0$ when $\beta = 0$ and $\beta = \pi$ for all t, we may assume the Fourier expansion

$$\psi(\alpha,\beta,t) = \sum_{n=1}^{\infty} f_n(\alpha,t) \sin n\beta \tag{23}$$

satisfying these conditions. Term-by-term differentiation with regard to β is justified (see e.g. Jeffreys and Jeffreys [14]). If this series is

substituted in (9) and we multiply by the general term sin nβ and integrate from $\beta = 0$ to $\beta = \pi$, we obtain the equations

$$f_n'' - n^2 f_n = r_n(\alpha,t) \ , \tag{24}$$

where

$$r_n(\alpha,t) = -\frac{2}{\pi}\int_0^{\pi}(\zeta/H^2)\sin n\beta \, d\beta. \tag{25}$$

The equations hold for $n = 1,2,3,\ldots$ and the primes denote differentiation with regard to α, i.e. $f_n' = \partial f_n/\partial\alpha$. Effectively we can treat these equations as ordinary differential equations. Although the time is present in the solutions it enters only through its presence in the quantities $r_n(\alpha,t)$. The integration of equations (24) is carried out at a fixed time and hence we can think of the solutions as dependent only on α at this fixed time.

The boundary conditions for equations (24) follow from (14) and (15). For all values of n

$$f_n = f_n' = 0 \quad \text{when } \alpha = \alpha^* \tag{26}$$

and, as $\alpha \to \infty$,

$$e^{-\alpha}f_n \to k\delta_n, \ e^{-\alpha}f_n' \to k\delta_n \tag{27}$$

where $\delta_1 = 1, \ \delta_n = 0 \ (n \neq 1)$.

These conditions can now be used along with the equations (24) to deduce a necessary condition on $r_n(\alpha,t)$ which holds for all values of the time.

Let

$$p_n(\alpha,t) = f_n' + nf_n. \tag{28}$$

Then

$$\frac{\partial}{\partial\alpha}\{e^{-n\alpha}p_n(\alpha,t)\} = e^{-n\alpha}r_n(\alpha,t)$$

and hence

$$e^{-n\alpha}p_n(\alpha,t) = \int_{\alpha^*}^{\alpha} e^{-nz}r_n(z,t)dz, \tag{29}$$

since $p_n = 0$ when $\alpha = \alpha^*$. Now let $\alpha \to \infty$ in (29). From (27) we find that

$$\left.\begin{aligned}\int_{\alpha^*}^{\alpha} e^{-nz}r_n(z,t)dz &= 2k, \text{ if } n = 1\\ &= 0 \ , \text{ if } n \neq 1 \ .\end{aligned}\right\} \tag{30}$$

Consider now how the result (30) may be used in the step-by-step time integration previously described. Let stage (i) be completed using, say, the scheme (19). Then ζ is known everywhere except on $\alpha = \alpha^*$ and $r_n(\alpha,t)$ can be found from (25) for corresponding values of α. If these values are substituted into (30), using a formula of numerical quadrature, this allows us to determine $r_n(\alpha^*,t)$ for any required value of n. In practice, of course, the upper limit in the integral in (30) is replaced by the finite value α_m corresponding to the boundary XY in Figure 3.

Having found $r_n(\alpha^*,t)$, the vorticity on the boundary $\alpha = \alpha^*$ can be found from the series which results from the inversion of (25), viz.

$$\zeta(\alpha,\beta,t) = -H^2 \sum_{n=1}^{\infty} r_n(\alpha,t) \sin n\beta. \qquad (31)$$

We thus succeed in determining ζ at $\alpha = \alpha^*$ without explicitly integrating the equations (24), i.e. stage (iii) is completed before stage (ii). The essential point now is that since $r_n(\alpha,t)$ is known for all stations α, including $\alpha = \alpha^*$, the equations (24) can be integrated by initial-value techniques with (26) as initial conditions. From the computed f_n we can calculate $\psi(\alpha,\beta,t)$ and we are then ready to enter stage (i) again.

The treatment of the closed cell problem is similar. In order to preserve comparability with the analysis just given we shall take the specific dimensions $OZ = \pi$, $OX = \ell$ in Figure 1. With the change in variables of x for α, y for β and with $H = 1$, equations (23), (24) and (25) hold good. The second and last of the boundary conditions (6) now give rise to the conditions

$$f_n = f_n' = 0 \quad \text{when } x = 0,\ell \qquad (32)$$

for the functions $f_n(x,t)$. Equation (29) holds if we put $\alpha = x$, $\alpha^* = 0$ and now, from (32), $p_n = 0$ when $x = \ell$ and hence

$$\int_0^{\ell} e^{-nz}\, r_n(z,t)dz = 0 \ . \tag{33}$$

Let us now introduce the function

$$q_n(x,t) = f_n' - nf_n. \tag{34}$$

We easily deduce the equation

$$e^{nx}q_n(x,t) = \int_0^{x} e^{nz}r_n(z,t)dz, \tag{35}$$

using the equations (24) and the condition that $q_n = 0$ when $x = 0$. Finally, since $q_n = 0$ when $x = \ell$, then

$$\int_0^{\ell} e^{nz}r_n(z,t)dz = 0 \ . \tag{36}$$

The two equations (33) and (36) serve the same purpose in this problem as the single equation (30) in the previous problem. For given n, each may be expressed as a formula of numerical quadrature. Then, given $r_n(x,t)$ for every station x except $x = 0$ and $x = \ell$, two simultaneous equations are obtained to determine $r_n(0,t)$ and $r_n(\ell,t)$. With this additional information it is possible to solve the equations (24) using step-by-step methods.

In practice several new points occur in this problem. One is that $\zeta(x,0,t)$ and $\zeta(x,\pi,t)$ are nonzero and, further, they will be unknown at the stage when $r_n(x,t)$ is evaluated. If conventional methods of numerical quadrature are used to evaluate $r_n(x,t)$, no additional problem occurs. The integrand of the integral in equation (25) is zero at the end points and the values of ζ at these points do not affect the integration. However, it is found that specialized formulae have to be used for the integration and this does create difficulties in the problem of flow in the closed cell. We shall return to this later. Another point in this problem is that it is not possible to use the equation (31) to determine ζ on the boundaries $y = 0$ and $y = \pi$ even when $r_n(x,t)$ is known. The series on the right does not converge to the function on the left at these points unless ζ is zero

there, i.e. we get the Gibbs phenomenon. It would be surprising if it were possible because we have not yet utilized the slope boundary conditions for ψ on $y = 0,\pi$. This point also will briefly be considered later.

NUMERICAL ANALYSIS

Two problems must be considered, the evaluation of the integral in (25) just mentioned, and the solution of the equations (24) by step-by-step methods. Each presents some difficulty if treated by conventional methods and we shall consider them in turn.

The problem with the integral on the right side of (25) is that unless a very fine grid is used in the β(or y) direction the result of conventional numerical quadrature will be inaccurate when n is large and the periodic function has many zeros in the range. Filon [15] pointed this out and proposed special formulae for such cases. The principle is based on polynomial approximation to the nonperiodic part of the integrand only, rather than to the whole integrand, as would be done in conventional methods. Filon gives details when the approximating polynomial is a parabola over three successive equally-spaced values of the integrand. For the integral on the right side of (25) the result, for integration over the points 2, 0, 4 shown in Figure 4, is

$$\int_{\beta_4}^{\beta_2} \zeta \sin n\beta \, d\beta \approx \frac{1}{n}(\zeta_4 \cos n\beta_4 - \zeta_2 \cos n\beta_2) + \frac{1}{2h_1 n^2}\{(3\zeta_4 - 4\zeta_o + \zeta_2)\sin n\beta_4 + (\zeta_4 - 4\zeta_o + 3\zeta_2)\sin n\beta_2\} - \frac{1}{h_1^2 n^3}(\zeta_4 - 2\zeta_o + \zeta_2)(\cos n\beta_4 - \cos n\beta_2) \, . \qquad (37)$$

The integral over the range $\beta = 0$ to $\beta = \pi$, assuming an even number of

intervals, is obtained by summing this result.*

Filon integration allows us to evaluate $r_n(\alpha,t)$ with good accuracy when n is large but it does create the difficulty that the function ζ needs to be known at the end points. This presents a problem only in the case of the closed cell flow. We shall consider this point later.

Next consider the solution of the equations (24) using step-by-step methods. It is convenient to drop the subscripts and also to consider the solutions as functions of a single variable, say x, i.e. we consider them at fixed time. Thus we consider the single equation

$$f'' - n^2 f = r(x) \tag{38}$$

and, to start, consider the solution defined in the finite range x = 0 to x = ℓ with the conditions (corresponding to (32))

$$f(0) = f'(0) = 0 \tag{39a}$$

$$f(\ell) = f'(\ell) = 0 \ . \tag{39b}$$

We assume r(x) to be given at all grid points on the interval x = 0 to x = ℓ, including the end points. At this stage also we shall abandon the notation of the points in Figure 4 and suppose that a typical grid point in the x-direction is denoted by $x_m = mh$ (m = 0,1,2,...) and that f_m denotes $f(x_m)$.

In theory, only the initial conditions (39a) are necessary to integrate (38) as a step-by-step problem. However, most step-by-step procedures applied to this problem are unsatisfactory, particularly if n is large. For example, the direct second-order approximation using central differ-

---

* For further details of this result, including the error term, see the National Bureau of Standards Handbook of Mathematical Functions (seventh Printing, May 1968) p. 890 together with the footnote, p. 890. The form we give in equation (37) is equivalent to Filon's actual form, but expressed differently.

ences gives the approximation

$$f_{m+2} = h^2 r_{m+1} + (2 + n^2h^2)f_{m+1} - f_m . \qquad (40)$$

Assuming some starting procedure which gives f_1, equation (40) can be used to construct an approximate solution starting from m = 0, but an elementary error analysis (see practically any textbook dealing with this subject) indicates that the error propagation is unsatisfactory. The error at some fixed value of x is unbounded as $h \to 0$ for all values of n. Also, for fixed h the error at fixed x increases rapidly with n. In the present application it is obvious that h will remain fixed regardless of n so that this latter property makes the use of a formula such as (40) unsuitable. To some extent the error growth is associated with the presence of the increasing exponential term exp(nx) in the complementary function of (38). This term plays little part in the required solution (in the cylinder problem it obviously, from the conditions (27), plays no part if n > 1) so the error growth is unacceptable. For the same reason any attempt to express the equation (38) as two simultaneous first -order equations with known initial conditions leads to a basically unsatisfactory system.

As an example, consider the pair of first-order equations defined by using the functions given by equations (28) and (34). The function p(x) (using the notation of the present section) satisfies the equation

$$p' - np = r(x) \qquad (41)$$

and q(x) satisfies

$$q' + nq = r(x) . \qquad (42)$$

The initial conditions are p(0) = q(0) = 0. If we express the first derivatives by simple forward differences, an approximation to (41) (the Euler method) is

$$p_{m+1} = (1 + nh)\, p_m + hr_m \tag{43}$$

and to(42)

$$q_{m+1} = (1 - nh)\, q_m + hr_m \ . \tag{44}$$

The error growth associated with (43) and (44) can be discussed following the analysis given by Forsythe and Wasow [16] . That associated with (43) will be unacceptable, but that associated with (44) may be acceptable. The reason is that the unwanted increasing exponential part of the complementary function of (38) has been isolated in (41) and the decreasing part in (42).

In the present application, however, we not only have initial conditions for $p(x)$ and $q(x)$ but also the conditions $p(\ell) = q(\ell) = 0$, i.e. more conditions than are theoretically needed. With the aid of these a satisfactory procedure can be constructed. Equation (42) is integrated in the increasing x-direction with initial condition $q(0) = 0$. Equation (41) is integrated backwards with $p(\ell) = 0$. An equivalent way of looking at the latter integration is to put $x = \ell - x'$ in (41) whence it becomes

$$dp/dx' + np = -r(\ell - x'), \tag{45}$$

with initial condition

$$p = 0 \text{ when } x' = 0. \tag{46}$$

Equation (45) is now of the same character for increasing argument as equation (42) and the error growth is at least tolerable.

From the functions $p(x)$ and $q(x)$ computed in this way, $f(x)$ and $f'(x)$ are obtained from the equations

$$\left.\begin{aligned} f &= (p - q)/2n \\ f' &= (p + q)/2 \ . \end{aligned}\right\} \tag{47}$$

Also, a highly sensitive check on the numerical procedure can now be made. It is that f and f' computed from (47) must now come out to be zero

(within an acceptable numerical tolerance) at both $x = 0$ and $x = \ell$. Moreover, this cannot possibly happen unless the conditions (33) and (36) have been correctly satisfied. The check therefore tests this in addition to the accuracy of the step-by-step process; it is a very severe one.

It is now clear from the above that we may restrict further consideration to the question of constructing an approximation to equation (42) with the initial condition $q(0) = 0$. A detailed error analysis of the approximation (44) indicates that, although the approximation is satisfactory in a manner not shared by the approximation (43), it cannot be expected to be effective unless nh is small compared with 1. We certainly cannot expect to obtain an accurate solution unless this condition is satisfied. Since h is fixed and n may be large this will not in general be satisfied. A method must be constructed which does not, on the whole, lose accuracy as n increases. Such a method is suggested by the principle of Filon quadrature.

We denote as before the value $x = mh$ by x_m and the corresponding value $q(x_m)$ by q_m. Equation (42) may be written

$$\frac{d}{dx}\{e^{nx}q(x)\} = e^{nx}r(x)$$

whence

$$q(x) = e^{-n(x-x_m)}\, q_m + e^{-nx}\int_{x_m}^{x} e^{n\xi}r(\xi)\, d\xi. \qquad (48)$$

If $x = x_m + kh$, this formula is a multi-step formula for the step-by-step integration of equation (42). The integral extends over k steps and we suppose in the usual way that $r(x)$ is approximated by a polynomial $P(x)$ passing through some, or all, of the corresponding $k+1$ points and perhaps also through points outside the range. Let the error of this approximation be

$$E(\xi) = r(\xi) - P(\xi)\ . \qquad (49)$$

Let $q^*(x)$ satisfy (48) when $r(x)$ is approximated by $P(x)$ and let

$$\varepsilon(x) = q(x) - q^*(x) \tag{50}$$

denote the error. Then

$$\varepsilon(x) = e^{-n(x-x_m)}\,\varepsilon_m + T(x)\ , \tag{51}$$

where $T(x)$ is the truncation error and is given by

$$T(x) = e^{-nx}\int_{x_m}^{x} e^{n\xi}\,E(\xi)\,d\xi\ . \tag{52}$$

Various estimates of $T(x)$ can be made, for example, a crude estimate is obtained by assuming only that $r(x)$ is continuous on the interval (x_m,x). By definition $E(x)$ is continuous on the same interval and thus

$$T(x) = E(\xi_1)\,\{1 - e^{-n(x-x_m)}\}/n, \tag{53}$$

where $x_m < \xi_1 < x$. Thus $T(x)$ cannot exceed the maximum error of the interpolating polynomial for any value of n.

Only the cases $k = 1$, $k = 2$ will be considered in detail as these are of the most value in practical integrations. The case $k = 1$ gives a one-step integration. Equation (51) becomes

$$\varepsilon_{m+1} = e^{-nh}\,\varepsilon_m + T_{m+1}, \tag{54}$$

where the last term denotes $T(x_{m+1})$. The solution is

$$\varepsilon_m = \gamma^m\varepsilon_o + S_m, \tag{55}$$

where

$$\gamma = e^{-nh}$$

$$S_m = \sum_{j=1}^{m} \gamma^{m-j}T_j.$$

If $\hat{T}$ is the magnitude of the numerically greatest of T_j then

$$|S_m| < \hat{T}(1 - \gamma^m)/(1 - \gamma). \tag{56}$$

From equation (55) we have the following simple properties of error propagation. For given h, $\gamma < 1$ and the error after m steps is bounded as $m \to \infty$. Moreover, as $h \to 0$ the error at a fixed value $x = mh$ remains bounded since $\gamma^m \to e^{-nx}$. For convergence, suppose x_o is the initial point $x = 0$ and that the initial condition (actually $q = 0$ in the present

application) has been calculated exactly. Then $\varepsilon_o = 0$ and convergence is established if $S_m \to 0$ as $h \to 0$ for fixed $x = mh$. Clearly $S_m \to 0$ if $\hat{T} \to 0$. From (53), for all m,

$$T_{m+1} \sim hE(\xi_1)$$

and $E(\xi_1)$ certainly remains bounded. Thus $\hat{T} \to 0$. Actually, as is well known, if P(x) passes through the j + 1 points

$$x_i, x_{i+1}, \ldots, x_{i+j}$$

and r(x) possesses continuous derivatives up to order j+1 on an interval which contains these points and the argument x, then

$$E(x) = (x - x_i)(x - x_{i+1})\ldots(x - x_{i+j})r^{(j+1)}(\eta)/(j+1)! \qquad (57)$$

where η lies on the given interval. In this case $E(\xi_1) = 0(h^{j+1})$ as $h \to 0$.

The formula for q* for the one-step case is

$$q^*_{m+1} = e^{-nh}q^*_m + e^{-nx_{m+1}} \int_{x_m}^{x_{m+1}} e^{n\xi}P(\xi)d\xi. \qquad (58)$$

The simplest form for $P(\xi)$ is to take it constant, i.e.

$$P(\xi) = r(x_m) = r_m.$$

This corresponds directly to the Euler formula in the approach using standard difference methods. The approximation obtained from (58) is

$$q^*_{m+1} = e^{-nh}q^*_m + r_m(1 - e^{-nh})/n \ . \qquad (59)$$

If we assume the expression (57) for the error, with i = m and j = 0 in this case, substitution in (52) yields

$$T_{m+1} = e^{-nx_{m+1}} \int_{x_m}^{x_{m+1}} (\xi - x_m)r'(\eta)\, e^{n\xi}d\xi \ . \qquad (60)$$

If we apply the mean value theorem then

$$T_{m+1} = e^{-nx_{m+1}}\, r'(\eta_1) \int_{x_m}^{x_{m+1}} (\xi - x_m)e^{n\xi}d\xi, \qquad (61)$$

where $x_m < \eta_1 < x_{m+1}$. We can deal with (61) in two ways, firstly by evaluation of the integral, which yields

$$T_{m+1} = \frac{r'(\eta_1)}{n}\{h - (1 - e^{-nh})/n\} . \qquad (62)$$

Alternatively, we can again employ the mean value theorem to give

$$T_{m+1} = \frac{1}{2} e^{n(\xi_1 - x_{m+1})} h^2 r'(\eta_1) , \qquad (63)$$

where $x_m < \xi_1 < x_{m+1}$. The two forms clearly become equivalent if, for fixed n, $h \to 0$. However, in general they display different properties. Equation (62) shows that the truncation error decreases with n. Equation (63) indicates that, regardless of n, the truncation cannot exceed $h^2 r'(\eta_1)/2$, which is what it would be if the Euler method were applied to integrate the basic differential equation (42) with $n = 0$.

A similar situation exists if P(x) is taken as the straight line joining x_m to x_{m+1}. The truncation error is then

$$T_{m+1} = \frac{1}{2!} e^{-nx_{m+1}} \int_{x_m}^{x_{m+1}} (\xi - x_m)(\xi - x_{m+1}) r''(\eta) e^{n\xi} d\xi$$

and it is easily shown that T_{m+1} cannot exceed in magnitude the term $-h^3 r''(\eta_1)/12$, where $x_m < \eta_1 < x_{m+1}$. This term gives the truncation error when trapezoidal integration is applied to the special case $n = 0$ of equation (42). We can clearly write down a formula of similar type for the truncation error when P(x) is a polynomial of any degree. However, the general investigation of this error is then more difficult. In particular, the term E(x) changes sign over the range of integration and the process of getting simple estimates of the kind just given is more involved. Although the point has not been investigated fully it does seem to be possible, following methods used by Steffensen [17] in the theory of numerical quadrature, to establish the following result. For

any given polynomial approximation P(x), the magnitude of the truncation error in approximating (48) does not exceed the magnitude of the corresponding truncation obtained when n is put equal to zero in this equation.

We turn finally to the actual formulae utilized in the numerical studies of this paper. Two formulae have been used, the first a one-step formula to effect the first step of the integration from the initial point x_o to the point x_1, but using parabolic approximation to r(x) over the points x_o, x_1, x_2. The second formula used was the two-step formula obtained by putting $x = x_m + 2h$ in (48) and using parabolic approximation for r(x) over the successive points x_m, x_{m+1}, x_{m+2}. The appropriate formulae for q* are as follows. For the first of these two

$$q_1^* = \gamma q_o^* + \frac{1}{n}(r_1 - \gamma r_o) - \frac{1}{2hn^2}\{r_2 - r_o - \gamma(4r_1 - 3r_o - r_2)\}$$
$$+ \frac{1}{h^2 n^3}(r_o - 2r_1 + r_2)(1-\gamma), \qquad (64)$$

where γ is as previously defined. For the second

$$q_{m+2}^* = q_m^* + \frac{1}{n}(r_{m+2} - \gamma^* r_m) - \frac{1}{2hn^2}\{(3r_{m+2} - 4r_{m+1} + r_m) - \gamma^*(4r_{m+1} - 3r_m - r_{m+2})\}$$
$$+ \frac{1}{h^2 n^3}(r_m - 2r_{m+1} + r_{m+2})(1-\gamma^*). \qquad (65)$$

Here $\gamma^* = e^{-2nh}$.

The resemblance between (65) and the Filon quadrature formula (37) is obvious.

The question of error propagation for two-step formulae such as (65) may be considered in a similar manner to that in the one-step case. The error equation is

$$\epsilon_{m+2} = \gamma^* \epsilon_m + T_{m+2}, \qquad (66)$$

where T_{m+2} is given by putting $x = x_{m+2}$ in (52). Actually the only new

point is that (66) has two independent solutions, one for even values of m generating from the initial value ε_0 and the other for odd values of m generating from ε_1. The general error propagation properties of these two solutions are essentially similar to those discussed in the one-step case and need not be considered further. In practice ε_1 will be determined by the starting approximation to q. This can be obtained using, for example, (64) or any other one-step formula. Generalization of these results to the multi-step case is obvious and we shall give no further discussion of this.

The truncation error T_1 in (64) is obtained by putting $m = 0$, $x = x_1$ in the formula (52). Also E(x) is given by (57) with $i = 0$, $j = 2$. The polynomial part of E(x) does not change sign over the range of integration so it is easy to deduce that $T_1 = e^{n(\xi_1-h)}h^4r'''(\eta_1)/24$, where $0 < \xi_1 < h$ and $0 < \eta_1 < 2h$. For the two-step formula (65), E(x) is given by putting $i = m$, $j = m+2$ in (57). In this case the polynomial part of E(x) does change sign over the range of integration. We have not investigated this case fully but, as previously noted, it is believed that the truncation can be shown to be not greater in magnitude then that obtained by putting $n = 0$ in (48). In this case this would give the truncation not greater in magnitude than that of the Simpson formula applied to the function r(x).

Finally we give some results of a specific numerical illustration of the whole computational procedure of solving equation (38) subject to the boundary conditions (39a) and (39b). Some computations were carried out to test the method. In this test r(x) was taken to be

$$r(x) = 100 + ae^{-x/2} + be^{-3x/2},$$

with a(n) and b(n) chosen to satisfy the necessary conditions (33) and (36). The range of integration was taken as $\ell = 1$. Equation (42) was

integrated in the positive x-direction and equation (41) in the negative x-direction as already described. Both integrations were carried out using formulae of type (64) and (65), equation (64) being used for the first step and (65) being used to continue the integration.

Computations were carried out for integer values of n from n = 1 to n = 20. For each case two approximate solutions were obtained using h = 0.1, 0.05 respectively and the results compared with the known exact solution. This comparison was found to be extremely satisfactory. In Table 1 some comparisons are given for one value only, the value f(0.5), but on the whole the result is representative of those obtained over the complete range. The values are given in floating decimal form.

n	Exact	h = 0.1	h = 0.05
1	1.87926E-01	1.87641E-01	1.87925E-01
5	1.19699E-01	1.19699E-01	1.19699E-01
10	5.65582E-02	5.65622E-02	5.65584E-02
15	3.04646E-02	3.04671E-02	3.04647E-02
20	1.86811E-02	1.86827E-02	1.86811E-02

TABLE 1. Values of f(0.5)

Apart from the very curious feature exhibited by the case n = 5, the evidence of this table is that the approximation improves as n increases; most certainly it does not get worse. We may summarize some other features of the numerical solutions as follows. For the case h = 0.05, the solutions computed using the step-by-step process agree with those computed from the exact solution to about 6 decimal places (not significant figures) at every grid point for every value of n = 1 to n = 20. In this case also, the check that f(0) and f(1) should both come out zero in the

final solution was satisfied to at least a tolerance of 10^{-5}, and more often 10^{-6} or 10^{-7}. When h = 0.1, this tolerance was rather greater, varying from 10^{-4} to 10^{-6}. In both cases this check improves as n increases, giving further evidence that the accuracy improves.

Finally, we must discuss briefly the solution of (38) in the case which corresponds to flow past a cylinder. Although, in practice, the range of integration is still restricted to be a finite number ℓ(corresponding to the imposed finite boundary XY in Figure 3), in theory this range is infinite in this problem. The new point is that the boundary conditions (39b) are not available now. Although, as we have already pointed out, only the conditions (39a) are in theory required to solve the step-by-step problem, obviously some replacement for (39b) must be found if the methods just described are to be used. No new problem exists for the determination of q(x) (note that x is used for α in this section). The condition q(0)=0 still holds. We do, however, require to know $p(\ell)$ to initiate the backward integration of (41).

To consider the problem in detail we should first examine the nature of the time-dependent solution of (10) for large α in order to ensure that the derivation of $r_n(a,t)$ from (25) is such that, as $\alpha \to \infty$, the integral in the condition (30) converges at its upper limit. This ensures that the problem is properly posed. Actually this could be done by linearizing the equation (10) with the outer boundary conditions (15), following the usual manner of Oseen, and obtaining the exact solution to this linearized problem. The problem has already been considered in the steady case ($\partial\zeta/\partial t = 0$ in (10)) for flow past a flat plate in the paper by Dennis and Dunwoody already cited. The treatment of the time-dependent case considered here is similar but much more involved and will not be considered. It will

be assumed, as is certainly the case in steady flow, that the behavior of $r_n(\alpha,t)$ as $\alpha \to \infty$ is such that the integral in (30) converges. To find an approximation to $p_n(\ell)$ at a finite ℓ we proceed as follows.

The complementary function of (41) involves the term exp(nx). From the boundary conditions (27) this term can only appear in the required solution when n = 1. Leaving aside this case, we write (41) in the form

$$np = p' - r \tag{67}$$

and then, for some large enough value $x = \ell$, take

$$p(\ell) \approx -r(\ell)/n \tag{68}$$

as an approximation to (67). This is a valid approximation provided that it makes $p'(\ell)$, viz. $-r'(\ell)/n$, small compared with $r(\ell)$. If this is satisfied, a second approximation is obtained from (67) in the form

$$p(\ell) \approx -\frac{1}{n}\{r(\ell) + r'(\ell)/n\} \tag{69}$$

provided that $r''(\ell)/n^2$ is small. The process may be continued as long as it remains valid. The case n = 1 differs only slightly. In this case it follows from (27) that the complementary function of (41) contributes a leading term. From the definition of p(x) this is seen to be $2ke^x$. The approximations (68) or (69) come from a particular solution which does not contain the complementary function. Thus when n = 1, the term $2ke^{\ell}$ must be added to the right side of (68) or (69). Actually the case n = 1 is not of much consequence since practically any standard step-by-step process will succeed in this case.

This method of approximating $p(\ell)$ is suggested by the fact that, in the case of steady flow (cf Dennis and Dunwoody),

$$r_n(x) \to nc, \text{ as } x \to \infty,$$

where c is a constant which can be expressed in terms of the drag exerted by the fluid on the cylinder. It is certain in this case that if ℓ is taken large enough the method will yield a valid approximation. In the

general time-dependent case the accuracy of the approximation must be judged according to the results achieved. Finally, one important factor influences the question of the accuracy of solutions obtained using approximations such as (68) and (69) to the initial condition for the integration of (41). The error-propagation of the backward integration of (41) is identical with that of the forward integration of (42). The particular discussions of equation (51) already given, and their obvious generalization, indicate that an error in the initial condition will rapidly damp out. Moreover, the higher the value of n, the more rapidly the damping will be. Thus errors introduced into $p(\ell)$ by approximations of the type considered rapidly lose their effect and there exists an inner range $0 < x < \ell'$, $(\ell' < \ell)$, where the accuracy is virtually independent of the approximation to $p(\ell)$.

SOME PRACTICAL DETAILS

We now deal with some points which have not yet been considered. For example, in the case of flow past a cylinder, the condition $\zeta = 0$ is a crude approximation to apply at the boundary XY (Figure 3) unless this is far from the axis $\alpha = \alpha^*$. It is valid in the early stages of the motion, before appreciable vorticity has had time to diffuse from the cylinder. At much later times, however, after the wake has built up behind the cylinder, it is found that vorticity of appreciable magnitude extends far downstream. The approximation $\zeta = 0$ on the boundary XY is then a poor one.

If we assume that for large time the flow tends to a steady state, the limit flow is found by putting $\partial\zeta/\partial t = 0$ in (10) and then $\zeta \equiv \zeta(\alpha,\beta)$. For large values of α, the conditions (15) are applicable. Thus the limit form of the equation (10) for t and α both large can be written

$$\nabla_1^2 \zeta = Rke^{\alpha}(\cos\beta \frac{\partial \zeta}{\partial \alpha} - \sin\beta \frac{\partial \zeta}{\partial \beta}) \ . \qquad (70)$$

A complete solution of this equation which satisfies the necessary conditions that $\zeta = 0$ when $\beta = 0$ and $\beta = \pi$ can be verified to be

$$\zeta(\alpha,\beta) = e^{\chi\cos\beta} \sum_{n=1}^{\infty} A_n K_n(\chi) \sin n\beta, \qquad (71)$$

where

$$\chi(\alpha) = Rke^{\alpha}/2$$

and K_n is the modified Bessel function of the second kind. The constants $A_n (n = 1,2,3,\ldots)$ are arbitrary.

Since χ is large when α is large and the leading term of the asymptotic expansion of $K_n(\chi)$ for large χ is

$$K_n(\chi) \sim (2/\pi\chi)^{\frac{1}{2}} e^{-\chi},$$

which is independent of n, then for large α equation (71) can be replaced by

$$\zeta(\alpha,\beta) \sim G(\beta)\ e^{\chi(\cos\beta - 1)-\alpha/2}\ , \qquad (72)$$

where $G(\beta)$ is a function of β alone. Equation (72) gives the character of the steady-state vorticity distribution for large α, viz. it is exponentially small except in a region near $\beta = 0$ which becomes narrower as α increases.

If the flow is steady, equation (72) can be used as a valid boundary condition (assuming α large enough) on the boundary XY in Figure 3. If $\alpha = \alpha_m$ denotes this boundary then (72) gives

$$\zeta(\alpha,\beta) \lessapprox \zeta(\alpha_m,\beta)\ \exp\{(\chi - \chi_m)(\cos\beta - 1)-(\alpha - \alpha_m)/2\}\ . \qquad (73)$$

Here χ_m denotes the value of χ at $\alpha = \alpha_m$. Actually the treatment given here is essentially similar to that described in a recent paper by Dennis, Hudson and Smith [18] in a problem of steady heat convection, although the details are not precisely the same.

For the general time-dependent case we cannot assert that (73) is

strictly valid owing to the presence of the nonzero $\partial\zeta/\partial t$ term in (10). However, the following argument is reasonable. If the solution approaches a steady state then (73) is a good approximation for moderate and large times, since $\partial\zeta/\partial t$ will be small. In the earlier stages of the motion the vorticity in the external field is anyway small. In any intermediate stage, the use of (73) can hardly be worse than assuming $\zeta = 0$, for it gives the general exponential-type decay which is inherent in the problem. Thus at a given time t, (73) may be assumed to give $\zeta(\alpha_m+h,\beta,t)$ from $\zeta(\alpha_m,\beta,t)$ for all values of β corresponding to the grid points on XY. Then (19) may be used to calculate $\zeta(\alpha_m,\beta,t + \Delta t)$ on XY.

One other point which requires some explanation is the question of how the solution of the problem of convection in the closed cell may be completed. It has already been noted that some difficulties arise in this problem, especially if the Filon quadrature formula (37) is used to calculate $r_n(x,t)$. It must be admitted that the method is less satisfactory in this case, but we may proceed as follows. Suppose the solution everywhere has been completed through time t. Equation (19) then gives $\zeta_o(t + \Delta t)$ at every internal grid point. Normally Δt is small to ensure stability of the process, so the solution for $\zeta_o(t + \Delta t)$ is not radically different from $\zeta_o(t)$. The new internal solution is then used with the old values (at time t) of ζ on $y = 0,\pi$ to evaluate $r_n(x,t + \Delta t)$. The equations (24) can now be solved and the solution for $\psi(x,y,t + \Delta t)$ at all internal grid points computed from (23). The solution will not be exactly right because we have used $\zeta_o(t)$ on $y = 0,\pi$. It will, however, be a very good approximation and it can now be introduced into the finite-difference equations (18) and improved using one of the usual iterative schemes. The iterative solution of the Poisson equation (3) is not entirely eliminated

in this problem, but is greatly reduced. From the final solution, boundary values of ζ on the boundary OZ (Figure 1) are calculated from (22) with similar equations on the other boundaries. This avoids the use of (31) and also ensures that the slope boundary condition is utilized on all the boundaries.

CALCULATIONS OF THE FLOW PAST A CIRCULAR CYLINDER

To illustrate the method we give brief details of some calculations of the flow past a circular cylinder. The details of the transformation (8) are those given by the equations (11) and (12). The dimensionless quantities ψ and ζ are related to the corresponding dimensional quantities ψ' and ζ' by the equations $\psi' = Ua\psi$, $\zeta' = (U/a)\zeta$. Here a is the radius of the cylinder. The Reynolds number in (10) is $R = Ua/\nu$, where U is the constant stream velocity. Actually it is more usual to work in terms of the Reynolds number $R^* = 2Ua/\nu$, and we shall adhere to this. The motion was started impulsively from rest and the calculations continued until a steady state was reached.

For a given Reynolds number there are several parameters which can be varied. Since the object of the calculations is to illustrate the method, some of these have been chosen to be consistent with those used in previous studies in the literature. This applies to the spatial grid sizes h and h_1, the time step Δt, and the value α_m defining the position of the boundary XY. In the present method the finite number n_o of terms taken to approximate the infinite series (23) is also a parameter. For each Reynolds number, separate calculations were carried out for at least two values of n_o. The details of the parameters used in the calculations are given in Table 2.

R*	Δt	$h=h_1$	α_m/π	n_o
7	0.03	$\pi/20$	1	5,10
10	0.04	$\pi/20$	1	5,10
20	0.08	$\pi/20$	1	5,10
40	0.04	$\pi/40$	1	5,10,20

TABLE 2. Parameters of calculations

The influence on the solutions of the number n_o given in this table can be summarized as follows. For each of the cases R* = 7,10,20 there was some change in the features of the solution from $n_o = 5$ to $n_o = 10$ but it was not great, although perhaps rather greater for R* = 20. For R* = 40 there was a somewhat greater change from $n_o = 5$ to $n_o = 10$ but a relatively insignificant one from $n_o = 10$ to $n_o = 20$. For this latter reason solutions for $n_o = 20$ were not obtained for the lower Reynolds numbers. Actually there is little doubt that as the Reynolds number increases, more terms of (23) are required to represent the stream function accurately, particularly for the late time and steady solutions, when the wake has fully developed. This is essentially the reason for the inadequacy of the higher Reynolds number solutions given by Dennis and Shimshoni [19] in a study of the steady flow problem. In this study a similar series to that used in the present paper was used for the stream function and also a series was adopted for the vorticity.

Most previously published material is concerned with the steady flow and we shall give details only of the final steady-state solutions obtained for the greatest value of n_o used in each case. In Figures 5,6,7 and 8 the steady streamlines are shown for the four cases. All patterns are reasonably consistent with previously known results. Figure 6 is consistent

with the results of Keller and Takami and Figure 8 with those of Kawaguti [20] and of Apelt. Actually the length of the separated region in Figure 8, measured from the rear generator, is about 2.5 diameters. Apelt obtained 2.13 diameters and Kawaguti about 2 diameters for this quantity in their investigations of steady flow. Kawaguti and Jain give about 2.76 diameters in their limiting solution for time-dependent flow. For the angle of separation at $R^* = 40$ we obtain about 54°. Estimates in the three investigations cited vary from 50° to 53.7°. Our value of 29° for the angle at $R^* = 10$ seems to agree well with Keller and Takami's result. The distribution of vorticity over the surface of the cylinder is shown for the four cases in Figure 9. Again the results seem to be in agreement with previous results, where comparisons are possible.

Distributions of pressure over the surface of the cylinder are shown in Figure 10. The quantity plotted is the dimensionless pressure coefficient

$$C(\beta) = \frac{p(\beta)-p_\infty}{\frac{1}{2}\rho U^2}, \tag{74}$$

where $p(\beta)$ is the pressure at given angle β on the surface and p_∞ is the constant pressure at large distances from the cylinder. According to Roshko [21], the coefficient $C(0)$ at the rear stagnation point should, for high enough Reynolds number, vary according to the law

$$C(0) \sim AR^{*-\frac{1}{2}}, \tag{75}$$

where A is constant. From the present solutions we obtain the results indicated in Table 3.

R^*	7	10	20	40
$C(0)$	-0.913	-0.770	-0.577	-0.528
A	-2.41	-2.43	-2.58	-3.34

TABLE 3. Pressure coefficient at $\beta = 0$

The quantity A in this table does not appear to be approaching a definite constant as R* increases, which we might expect if the result (75) were valid. However, perhaps the Reynolds numbers are a little small for any definite conclusions to be made.

The drag force D(t) exerted by the fluid on the cylinder is usually expressed in terms of a dimensionless drag coefficient $C_D(t)$ defined by

$$C_D(t) = D(t)/\rho U^2 a.$$

It can easily be shown in the present case of a circular cylinder (see e.g. Kawaguti and Jain [8] that

$$C_D(t) = -\frac{4}{R^*}\int_0^\pi \zeta_0 \sin\beta d\beta + \frac{4}{R^*}\int_0^\pi (\partial\zeta/\partial\alpha)_0 \sin\beta d\beta, \qquad (76)$$

where the subscript here denotes values at $\alpha = 0$. The first term on the right gives the friction drag coefficient and the second the pressure drag coefficient, denoted respectively by C_{Df} and C_{Dp}. It may be observed that very simple expressions can be obtained for these quantities in the present method. From (31), noting that $H^2 = e^{-2\alpha}$, we obtain

$$C_{Df}(t) = 2\pi r_1(0,t)/R^*$$

and

$$C_{Dp}(t) = -2\pi\{r_1'(0,t) - 2r_1(0,t)\}/R^*,$$

where the prime denotes differentiation with regard to α.

Some values of the drag coefficients in the limit solution for large t are shown in Table 4. They are assumed to be those appropriate to the limit $t \to \infty$, since they had approximately converged to steady values.

R*	$C_{Df}(\infty)$	$C_{Dp}(\infty)$	$C_D(\infty)$
7	1.565	1.882	3.447
10	1.247	1.599	2.846
20	0.790	1.195	1.985
40	0.524	0.999	1.523

TABLE 4. Steady drag values

On the whole they agree well with previous estimates. Keller and Takami give the total drag coefficient as 2.74 at R* = 10. For R* = 40, Apelt gives $C_D(\infty) = 1.496$ and Kawaguti $C_D(\infty) = 1.618$.

The above selection of results indicate that, for the cases considered, results of accuracy comparable with those in the literature can be obtained. Actually solutions for R* = 70 and R* = 100 have also been attempted. Firstly, R* = 100 was attempted using the same grid sizes as for R* = 40. A steady state solution was obtained, but a distorted one which indicated quite unrealistic properties and which were obviously due to poor finite-difference approximations. The same would most surely have been true of the limiting solution at R* = 100 in Kawaguti and Jain's work, had they not discontinued the calculation at an early time, for they used a grid size of only $\pi/30$. A solution for R* = 70 was then attempted taking $h = h_1 = \pi/60$. This was successful and was clearly approaching a realistic solution for large times. A similar solution was then attempted for R* = 100 with $h = h_1 = \pi/60$. This, too, was successful. However, the approach to the final solution in the wake was so slow that both solutions were discontinued after a large number of time steps.*

* Both have since been completed using steady-state methods and will shortly be published.

SUMMARY AND CONCLUSION

A method has been described which allows certain classes of time-dependent integrations of the Navier-Stokes equations to be treated entirely by initial-value techniques. The main object has been to deal with the problem of flow past cylinders. One of the objects was the hope that the introduction of initial-value techniques would speed up the integrations. As yet it is too early to say how well this has been achieved, but one example can be quoted. Kawaguti and Jain give a typical time of 2 hours CDC 3600 machine time required to complete their study of the flow past a circular cylinder at $R^* = 40$, using grid sizes $h = h_1 = \pi/30$. A similar study on the same machine using present methods with the same h and h_1 (which are actually rather too large to give a very good solution at this Reynolds number) took less than half an hour. However, comparisons are difficult because, naturally, much depends upon the efficiency of the particular iterative method used to solve the Poisson equation if boundary-value techniques are used.

Actually, if steady-state solutions are the object of the integration, it has been found more satisfactory to use steady-state methods (i.e. solve (10) with $\partial\zeta/\partial t = 0$). It is quite easy to adapt the above method to this case. In a recent project carried out jointly at the Mathematics Research Center and the University of Western Ontario a number of problems involving steady flow past cylinders have been solved. These will be published subsequently.

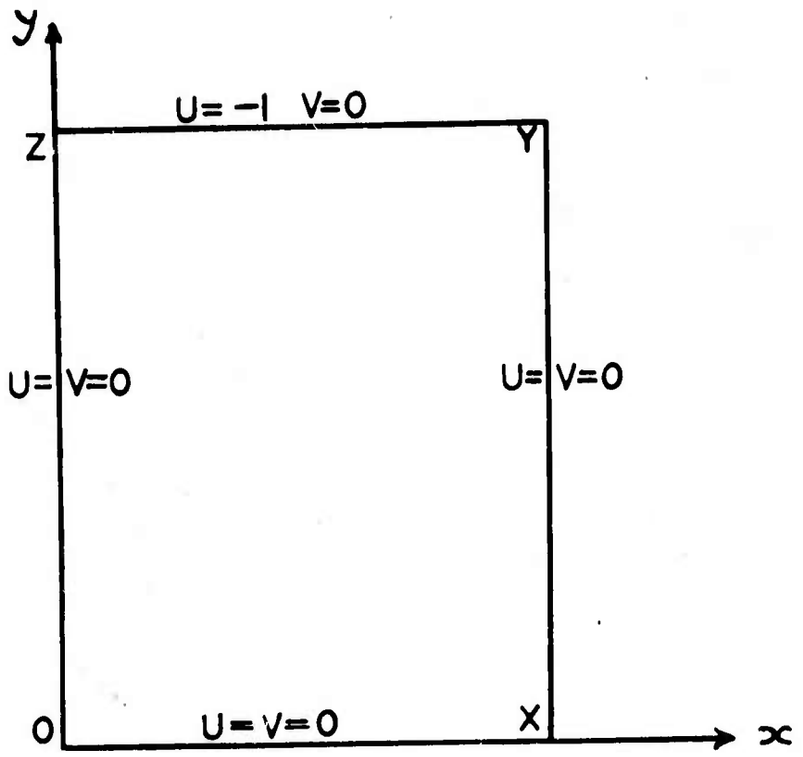


FIG.1

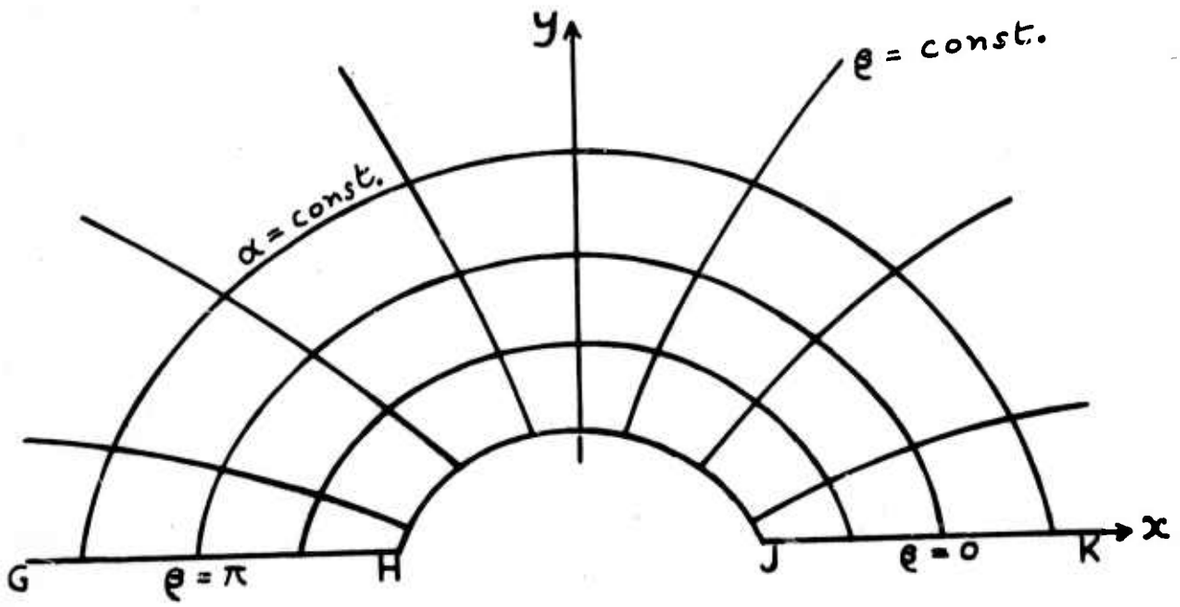


FIG.2

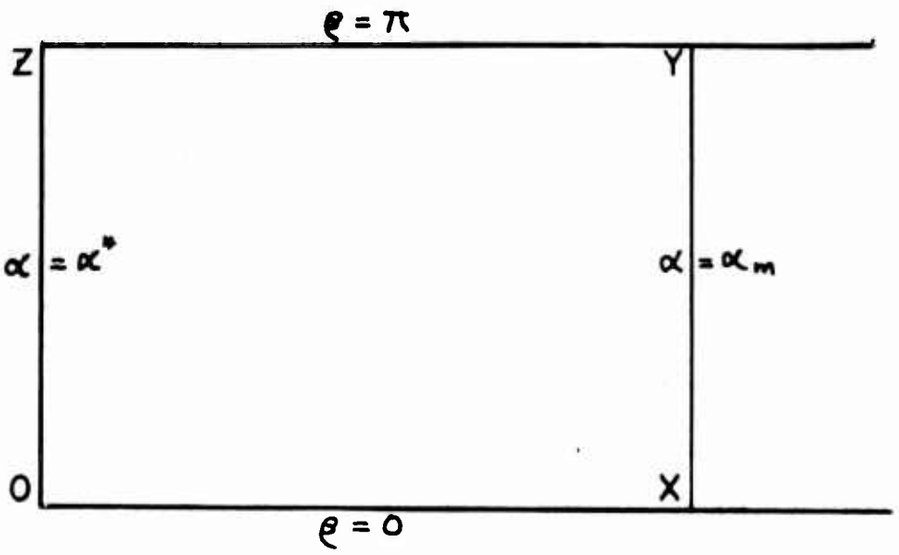


FIG.3

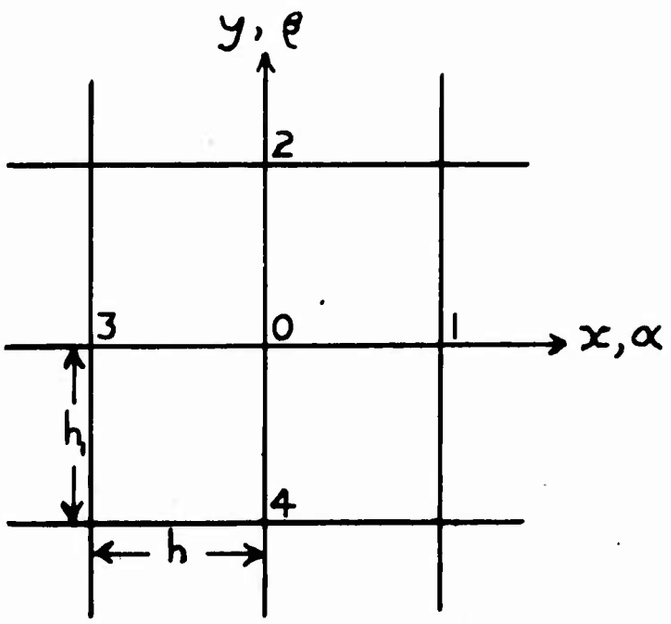


FIG.4

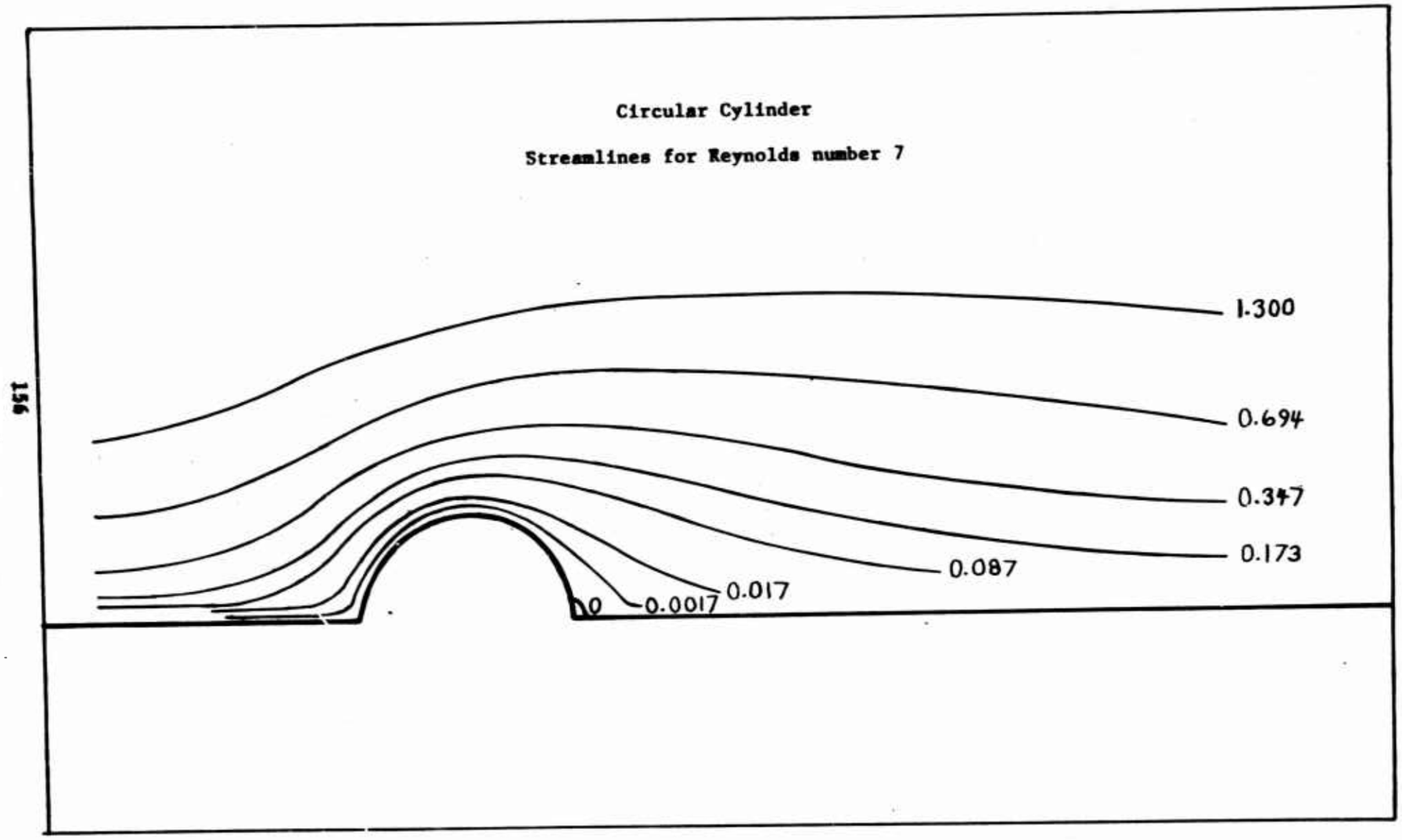

Circular Cylinder
Streamlines for Reynolds number 7
1.300
0.694
0.347
0.173
0.087
0.017
0.0017
0

FIG.5

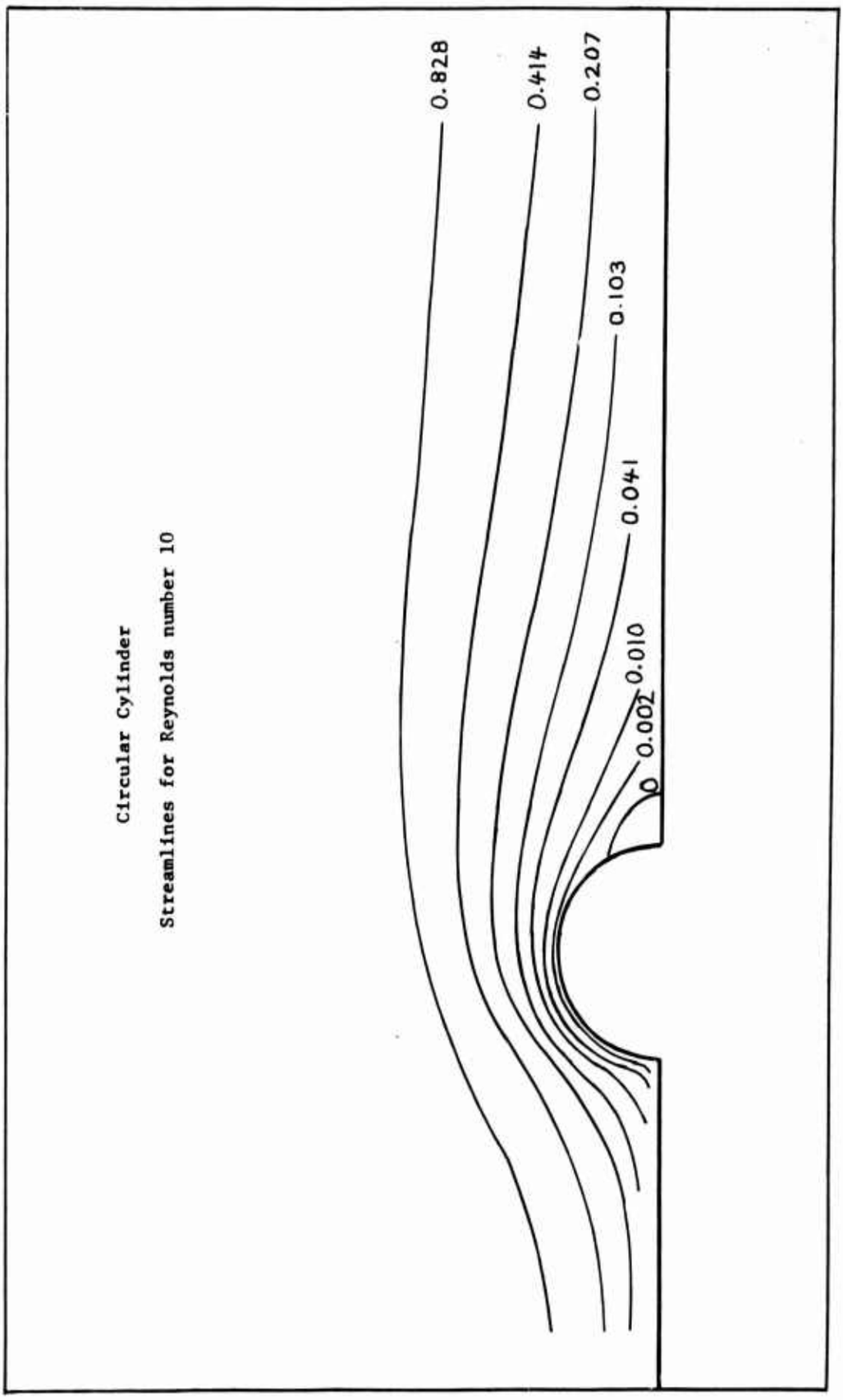

Circular Cylinder
Streamlines for Reynolds number 10
0.828
0.414
0.207
0.103
0.041
0.010
0.002
0

FIG.6

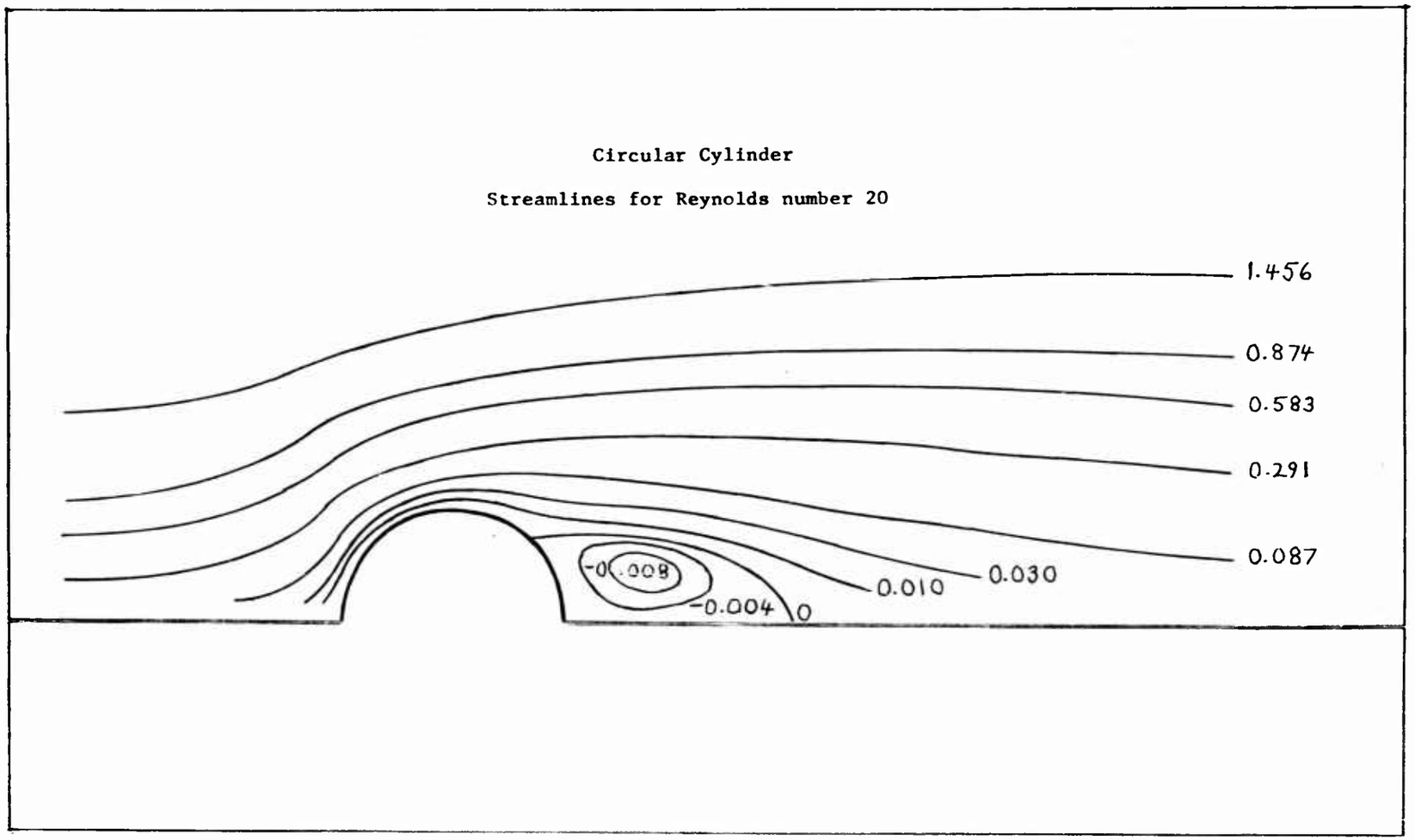

Circular Cylinder
Streamlines for Reynolds number 20
1.456
0.874
0.583
0.291
0.087
0.030
0.010
-0.004
0

FIG.7

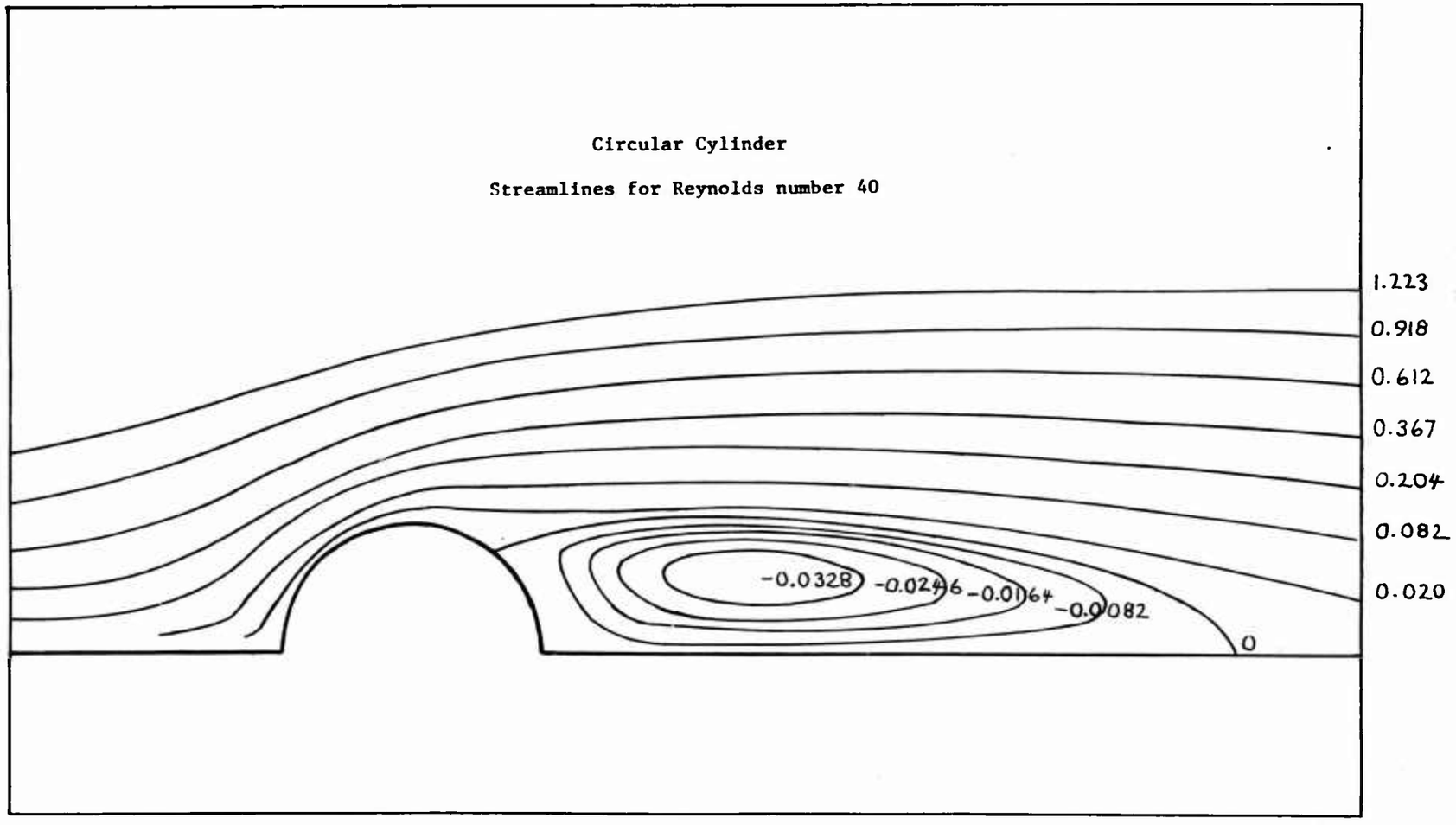

Circular Cylinder
Streamlines for Reynolds number 40
1.223
0.918
0.612
0.367
0.204
0.082
0.020
-0.0328
-0.0246
-0.0164
-0.0082
0

FIG.8

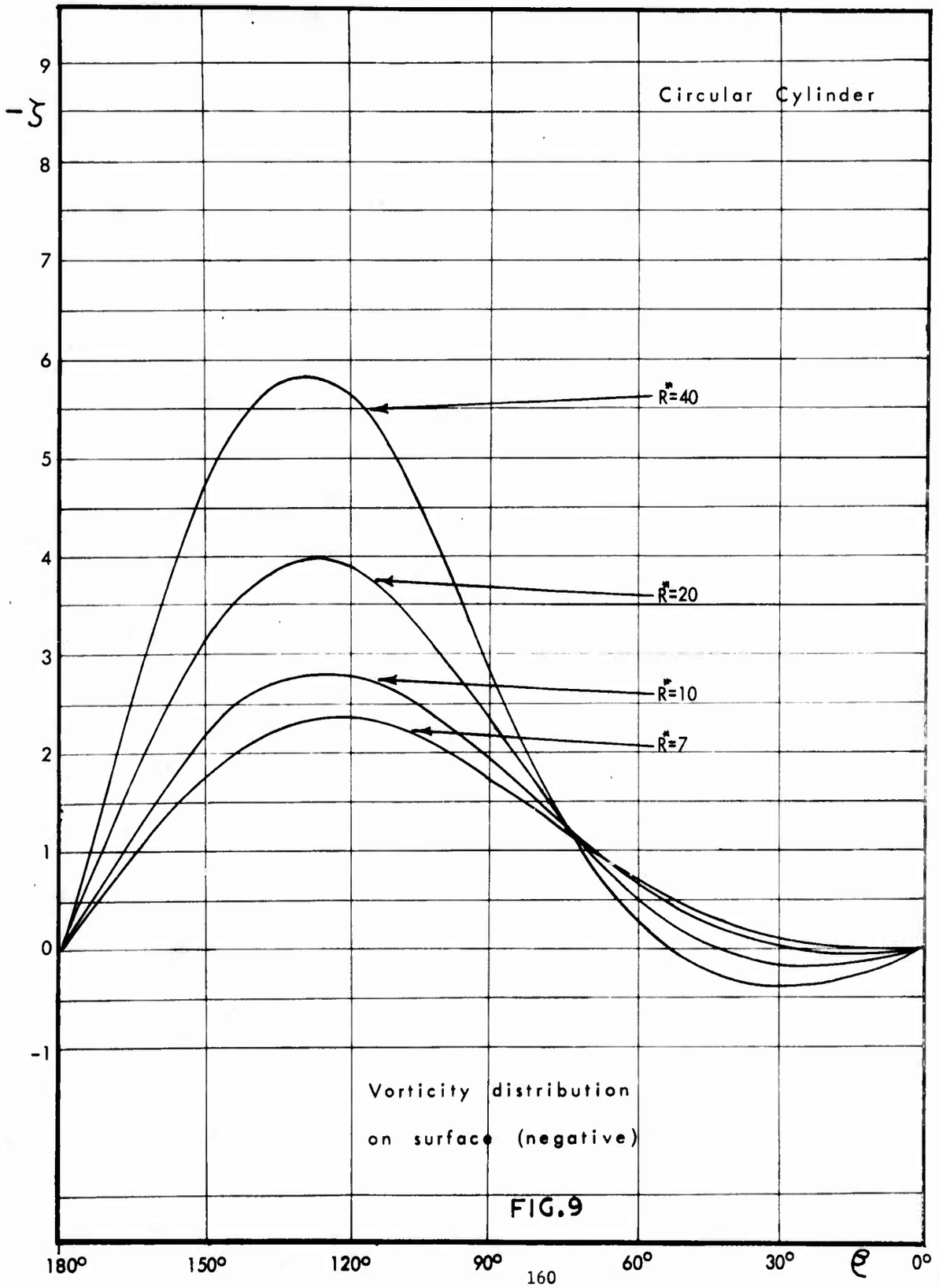

Circular Cylinder
$-\zeta$
9
8
7
6
5
4
3
2
1
0
-1
$R^*=40$
$R^*=20$
$R^*=10$
$R^*=7$
Vorticity distribution
on surface (negative)
180°
150°
120°
90°
60°
30°
0°
θ

FIG.9

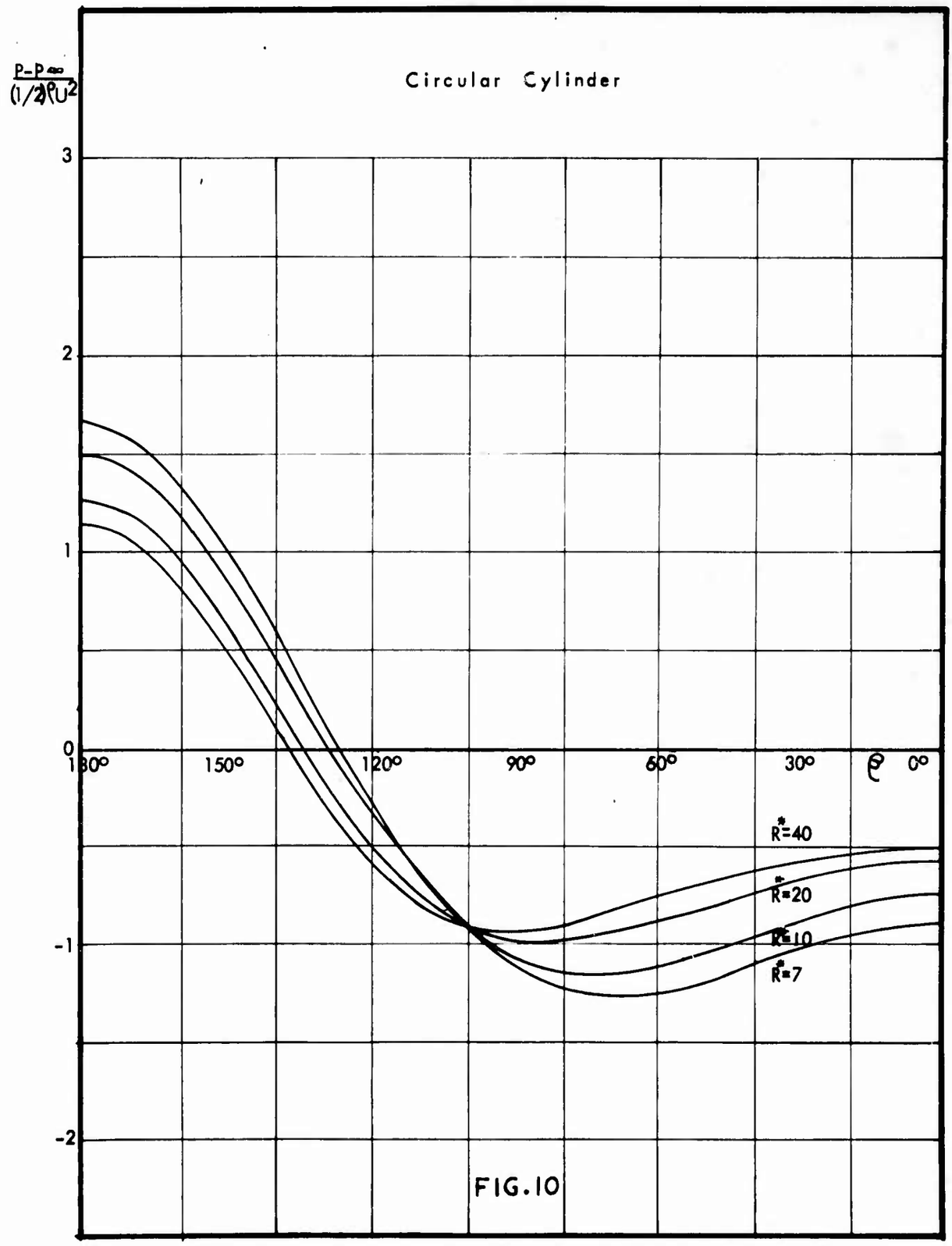

Circular Cylinder
$\frac{P-P_\infty}{(1/2)\rho U^2}$
3
2
1
0
-1
-2
180°
150°
120°
90°
60°
30°
0°
φ
R*=40
R*=20
R*=10
R*=7

FIG. 10

REFERENCES

1. D. Greenspan, P.C. Jain, R. Manohar, B. Noble and A. Sakurai. Numerical studies of the Navier-Stokes equations. Mathematics Research Center, U.S. Army, Technical Summary Report No. 482. Madison, 1964.

2. M. Kawaguti, J. Phys. Soc. Japan 16 (1961), 2307.

3. L.M. Simuni, Inzhenernii Zhournal (USSR) 4 (1964),446. (Translated by O.R. Burggraf.)

4. R.D. Mills, J. Roy. Aero. Soc. 69 (1965), 116, 714.

5. O.R. Burggraf, J. Fluid Mech. 24 (1966), 113.

6. F. Zabransky. Numerical solutions for fluid flow in rectangular cavities. Ph.D. Thesis, University of Western Ontario, 1968.

7. R.B. Payne, J. Fluid Mech. 4 (1958), 81.

8. M. Kawaguti and Padam Jain. Numerical study of a viscous fluid flow past a circular cylinder. Mathematics Research Center, Technical Summary Report No. 590. Madison, 1965.

9. C.J. Apelt. The steady flow of a viscous fluid past a circular cylinder at Reynolds numbers 40 and 44. Aeronautical Research Council Reports and Memoranda No. 3175. London, 1961.

10. H.B. Keller and H. Takami. Numerical solution of nonlinear differential equations (D. Greenspan, Ed.), John Wiley and Sons, New York, 1966, p. 115.

11. E. Janssen, J. Fluid Mech. 3 (1957), 329.

12. S.C.R. Dennis and J. Dunwoody, J. Fluid Mech. 24 (1966), 577.

13. L. Fox. The numerical solution of two-point boundary problems in ordinary differential equations. Clarendon Press, Oxford, 1957, p. 18.

14. H. Jeffreys and B.S. Jeffreys. Methods of Mathematical Physics. 3rd edn. Cambridge University Press, 1962, p. 441.

15. L.N.G. Filon, Proc. Roy. Soc. Edin. 49 (1928), 38.

16. G.E. Forsythe and W.R. Wasow. Finite-difference methods for partial differential equations. John Wiley and Sons, New York, 1960, p.30.

17. J.F. Steffensen. Interpolation. The Williams and Wilkins Company, Baltimore, 1927.

18. S.C.R. Dennis, J.D. Hudson and N. Smith, Phys. Fluids 11 (1968), 933.

19. S.C.R. Dennis and M. Shimshoni. The steady flow of a viscous fluid past a circular cylinder. Aeronautical Research Council Current Papers No. 797. London, 1965.

20. M. Kawaguti, J. Phys. Soc. Japan 8 (1953), 747.

21. A. Roshko. A review of concepts in separated flow. Proc. Canadian Congress of Applied Mechanics, Centennial Year, Vol. 3, 1967, p. 81.

ATTENDANCE LIST

Lionel J. Adams, U.S. Army Aviation Materiel Command
Raymond Bellucci, Atmospheric Sciences Lab, Fort Monmouth
Harold J. Breaux, Aberdeen Proving Ground
Robert J. Brown, Picatinny Arsenal
Aivars Celmins, Aberdeen Proving Ground
Henry Chambers, USAECOM, Fort Monmouth
Choong Yun Cho, U.S. Army Watervliet Arsenal
Bennie D. Cooley, U.S. Army Aviation Materiel Command
Louis Paul Crnarich, Picatinny Arsenal
John D'Agostino, USAECOM, Fort Monmouth
S.C.R. Dennis, Mathematics Research Center
F.G. Dressel, U.S. Army Research Office-Durham
Sylvan Eisman, Frankford Arsenal
Alfred Farnochi, Fort Monmouth
Fred Frishman, OCRD, Washington, D.C.
John Giese, Aberdeen Proving Ground
Bruce C. Gray, Fort Detrick
Harry A. Greveirs, Frankford Arsenal
A.D. Hall, Bell Telephone Laboratories
John P. Hakim, Fort Monmouth
Morton A. Hyman, Mathematics Research Center
Donald F. Kass, U.S. Army Natick Laboratories
Sheldon D. Kramer, Fort Belvoir
Robert Lefrer, USAECOM, Fort Monmouth
Leon Leskowitz, USAECOM, Fort Monmouth
Henry Lisman, Fort Monmouth
Jeffrey M. Luce, Block Engineering, Inc., Cambridge, Mass.
Roger MacGowan, Washington Navy Yard
J. Matsushino, Fort Huachuca, Arizona
James J. McLaughlin, U.S. Army Materials & Mechanics Research Center
John Mescall, U.S. Army Materials & Mechanics Research Center
George Millman, USAECOM, Fort Monmouth
Ben Noble, Mathematics Research Center
Arnold O. Pallingston, Picatinny Arsenal
Leon Pancoast, Frankford Arsenal
Gene B. Parrish, U.S. Army Research Office-Durham
Hope S. Perlman, USAECOM, Fort Monmouth
William D. Powers, Pacific Missile Range, Point Mugu, California
Louis B. Rall, Mathematics Research Center
Gordon T. Sande, Princeton University
Jerrold M. Shapiro, USAECOM, Fort Monmouth
James T. Tracey, Picatinny Arsenal
Eileen D. Ulrich, USAECOM, Fort Monmouth
Andries van Dam, Brown University
Joseph Weinstein, USAECOM, Fort Monmouth
John R. Whiteman, Mathematics Research Center
J.M. Yohe, Mathematics Research Center

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* U.S. Army Research Office-Durham Box CM, Duke Station Durham, North Carolina 27706	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP NA

3. REPORT TITLE

Proceedings of the 1968 Army Numerical Analysis Conference

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Interim Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

6. REPORT DATE December 1968	7a. TOTAL NO. OF PAGES 165	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ARO-D Report 68-3 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

This document is subject to special export controls and each transmittal to foreign nationals may be made only with prior approval of the U.S. Army Research Office-Durham.

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Army Mathematics Steering Committee on behalf of the Office of the Chief of Research and Development

13. ABSTRACT

This is the technical report resulting from the 1968 Army Numerical Analysis Conference. It contains 5 papers, which treat various computer and computational problems of interest to the scientific world.

14. Key Words

ALTRAN
Hankel and Toeplitz Matrices
Fast Fourier transforms
Fourier processing shortcuts
numerical experiments on a Stefan problem
programming for accuracy
time sharing
computation of the dynamic response of membranes
Navier-Stokes equations
avionics systems simulator
computer graphics

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Unclassified
Security Classification